Étude d'algorithmes généraux d'optimisation appliqués à des problèmes de tomographie

par

Jonathan BAILLARGEON

mémoire présenté au Département de mathématiques
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, décembre 2018

Le 18 décembre 2018

Membres du jury

Professeur Jean-Pierre Dussault
Directeur de recherche
Département d'informatique

Professeur Maxime Descoteaux
Membre interne
Département d'informatique

Professeur François Dubeau
Président-rapporteur
Département de mathématiques

À ma famille et ami(e)s, sans qui rien ne serait pareil

SOMMAIRE

Dans ce mémoire, nous évaluerons l'application de techniques modernes d'optimisation sur un problème pratique qui possède déjà des techniques de résolution que nous croyons sous-optimales. Nous introduirons un algorithme récent, nous introduirons une adaptation de FISTA et nous donnerons plusieurs pistes de développement. De plus, nous profiterons de l'occasion pour tester le nouveau langage JuliaLang, que nous appellerons Julia pour le reste de cet ouvrage.

REMERCIEMENTS

Jonathan Baillargeon

Sherbrooke, juillet 2018

J'aimerais prendre ces quelques paragraphes pour remercier ceux qui m'ont permis de me rendre aussi loin. Tout d'abord, mon superviseur Jean-Pierre, qui a accepté de me donner la chance de vivre cette fabuleuse aventure. Les conseils culinaires, touristiques et vinicoles, mais surtout une très grande patience et une gentillesse hors du commun ne seront jamais oubliés.

Je veux remercier tout les gens du BISOUS (Bureau d'Imagerie, Simulation et Optimisation de l'Université de Sherbrooke) pour leur aide indispensable. Que ce soit Luc Marchand sur le plan théorique, Samuel Goyette et Adam Salvail pour leur humour, Émilie Joannopoulos pour sa bonne humeur et son aide, Tangi Migot pour son aide en Julia et les fabuleuses parties de Mulky et Mathieu Frappier pour les distractions. Travailler dans une atmosphère aussi stimulante et accueillante est une chance dont peu de gens puissent se vanter.

Je dois remercier Maxime Toussaint à part vu son énorme contribution. Tu m'as tellement aidé durant ces années, tu mérites amplement ton propre paragraphe.

Je veux remercier Clément Zotti pour son aide en C, François Rheault pour les discussions

au gym, Michael Paquette pour les discussions partout et Pier-Luc Audet pour son divan si peu confortable qui m'a sauvé plus d'une fois.

Je sais que j'oublie sûrement beaucoup de gens, que ce soit mes amis du babyfoot, des compétitions de Super Smash, des cours de danse, des dégustations de bières sures ou de tous les mathématiciens formidables que j'ai eu la chance de côtoyer durant mon parcours académique.

Finalement, je veux remercier les membres de ma famille, qui ont toujours été là pour moi et qui m'ont offert leur support jusqu'à la fin. Je ne serais pas l'ombre de ce que je suis aujourd'hui sans eux.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Sujet pratique, utilité dans notre société

La Tomographie par Émission de Positons (TEP), et la Tomodensitométrie (TDM), sont deux techniques d'imagerie nucléaire qui sont actuellement utilisées pour différentes tâches dans le milieu médical. Bien que ces méthodes exposent les patients à des doses de radiation non négligeables par rapport à d'autres techniques telles que l'Imagerie par Résonance Magnétique (IRM), ou bien les échographies, ces techniques offrent des avantages justifiant leur survie.

Les reconstructions des images en TEP et en TDM peuvent en fait être vues comme des problèmes mathématiques de minimisation. On parle donc d'algorithmes de reconstructions tomographiques. L'augmentation de l'efficacité des algorithmes pour résoudre ces problèmes est une priorité. Tout d'abord, un temps de reconstruction moindre de l'image permet d'avoir un diagnostic plus rapidement, ce qui entraîne un traitement plus adéquat, une meilleure utilisation des tomographes et des ordinateurs, etc. Ensuite, un algorithme capable d'utiliser moins de ressources lors de l'acquisition dans notre cas est un algorithme nécessitant moins d'expositions aux rayons ionisants pour les patients, ce qui est un avantage considérable, et l'une des préoccupations croissantes dans le milieu. Il faut différencier deux types de ressource. Les premières sont à surveiller pour le côté

algorithmique, et les autres le sont pour le côté modélisation. Nous appellerons les premières «ressources de calcul» et les autres seront désignées par «intensité de radiation». L'algorithme parfait serait minimal pour les deux cas, et lorsqu'il y a égalité entre plusieurs algorithmes dans un de ces cas, on voudrait alors dominer dans l'autre. Pour cet ouvrage, les ressources nécessaires lors de l'acquisition seront mesurées par le nombre de projections nécessaires, soit la taille de l'outil que nous définirons plus tard comme étant la matrice système.

La modélisation de ces problèmes ainsi que le fonctionnement de ces techniques d'imagerie seront expliqués dans le chapitre un. Les algorithmes de reconstruction quant à eux seront approfondis dans le chapitre deux. Le chapitre trois fera mention d'une méthode très récente adaptée pour le problème de reconstruction tomographique. Le chapitre quatre sera consacré à la méthodologie des modèles et des tests. Finalement, le chapitre cinq comprendra les différents tests et les résultats.

Deux annexes comprennent respectivement les détails d'un nouvel algorithme prometteur mais non complété, et les images intéressantes mais considérées trop lourdes pour le chapitre cinq.

Le sujet principal de ce mémoire est donc l'étude, l'analyse et la comparaison de différents algorithmes issus du monde de l'optimisation moderne, ainsi que d'un algorithme de reconstruction tomographique usuel pour la TEP à titre de comparaison. L'évaluation des algorithmes généraux sera faite pour la TDM et la TEP. C'est un sujet très appliqué, ayant une documentation fort impressionnante et dont je ne peux qu'effleurer la surface.

Désir de tester des méthodes récentes

L'apport de ce mémoire, pour la TEP, est de jeter un regard moderne sur les techniques déjà utilisées et de les comparer aux récentes avancées en optimisation. Principalement, l'application de FISTA, ainsi que du gradient conjugué à mémoire limitée de Hager et Zhang, tous deux respectivement mieux décrits au chapitre trois et en Annexe A. Il sera fait mention dans les futurs chapitres que les algorithmes utilisés doivent être capables de gérer des dimensionnalités immenses, puisqu'on aura une variable par PIxEL (PIcture ELement) ou par VOxEL (VOlume ELement). On utilisera donc des méthodes ne nécessitant pas de matrices hessiennes, soit des algorithmes d'ordre un. Avec l'importance croissante des problèmes de très grandes dimensions, il n'est pas étonnant de voir de nouvelles techniques apparaîtrent ici et là. Une analyse comme celle-ci aura avantage à être refaite à l'avenir. Pour faciliter une possible suite à ce mémoire, le code utilisé, en Julia, est disponible sur le Github suivant : https :github.com/JoBaill/TomoTools.git.

CHAPITRE 1

Tomographie

1.1 Qu'est-ce que la TOMOGRAPHIE

La tomographie est une technique d'imagerie par laquelle on explore les tranches d'un objet à l'aide d'une source d'intérêt afin de reconstruire son volume. Les méthodes de tomographie sont basées sur un examen médical qui ne nécessite aucune effraction de la peau autre que pour prélever du sang ou pour injecter un produit. Elles utilisent cependant des sources irradiantes qui peuvent affecter le sujet. Ce sont donc des méthodes qui sont nocives. Nous étudierons deux branches de la tomographie : La TDM et la TEP.

1.2 Tomodensitométrie (TDM)

La TDM est une technique d'imagerie médicale structurelle qui consiste à mesurer l'absorption des rayons X par le corps humain lorsque ceux-ci le traversent, et ainsi utiliser cette information afin de reconstruire une carte interne du corps, sans avoir à l'ouvrir.

L'acquisition consiste donc à mettre le corps humain entre une source ionisante et un panneau de détecteurs, puis mesurer l'atténuation de l'intensité entre la source (début) et les détecteurs (fin). La différence représente la quantité absorbée par le sujet, ce qui va permettre de retrouver la composition anatomique de celui-ci. En effet, on connaît le taux d'absorption des différents éléments se retrouvant dans le corps humain (eau, chairs, os, air, etc.). Si l'on discrétise le corps comme étant une carte (à l'aide de pixels en 2D, voxels en 3D), alors chaque pixel/voxel se voit assigner une valeur. Cette valeur représente le taux d'absorption du milieu que représente ledit voxel. On peut ensuite facilement étendre ces valeurs sur les différentes nuances de gris, et ainsi avoir une image en noir et blanc de l'intérieur du corps humain, et ce, sans avoir eu à l'ouvrir.

Les tomodensitomètres (CT-Scan) ont typiquement la forme d'un tore lorsque vu de l'extérieur, dans lequel est insérée une planche afin de soutenir le patient. On désignera maintenant le tore comme étant l'arceau (issus du terme gantry).

Figure 1.1 – CT-Scan vu de l'extérieur

Un théorème célèbre, démontré en 1917 par Johann Radon [Rad86], stipule qu'avec toutes

les projections d'un objet selon des droites concourantes, il est possible de reconstituer la fonction réelle à deux variables $f(x, y)$ représentant l'objet.

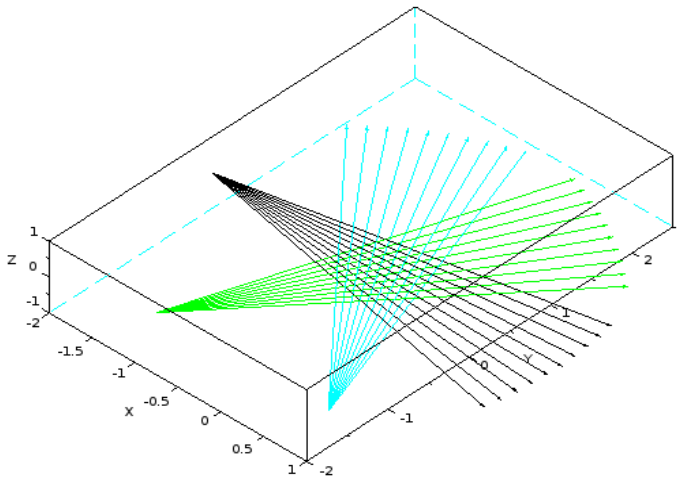


Figure 1.2 – Exemple de projections en TDM, en point : la source ionisante, aux extrémités : les détecteurs

La figure 1.2 représente la couverture effectuée par les rayons issus d'une source (la pointe) vers les détecteurs (extrémités des éventails). Ce couple source ionisante/panneau de détecteurs tourne à l'intérieur de l'arceau, ce qui procure des projections selon des droites concourantes sous plusieurs angles autour du corps du patient. Il existe plusieurs approches pour faire une acquisition en TDM, mais dans le cadre de ce travail, nous avons choisi le cas où les rayons ne sont pas parallèles entre eux. Après avoir fait toutes les projections voulues pour une position de l'arceau, la planche avance et le processus recommence. La planche pourrait avancer de manière continue, ce qui aurait pour conséquence que les positions de la source ionisante formeraient une hélice (hélicoïdale), mais pour ce travail, nous avons implémenté la version saccadée décrite précédemment.

Bien qu'il soit impossible en pratique d'obtenir l'entièreté des projections comme le voudrait le théorème, il est quand même possible d'en avoir suffisamment afin d'obtenir un résultat convenable pour l'utilisation clinique.

Pour modéliser l'interaction des rayons X dans un objet, on utilise le fait que l'atténuation d'un photon de ces rayons suit la Loi de Beer-Lambert, c'est-à-dire que l'intensité mesurée à l'arrivée est proportionnelle à l'intensité de départ, moins la dose perdue par l'absorption des environnements traversés. En utilisant la version discrétisée de cette loi, et en définissant les éléments suivants comme suit, on obtient :

- I_j = Intensité mesurée pour la projection j,
- I_0 = Intensité de départ (évaluée avec acquisition sans objet),
- μ_i = coefficient d'absorption pour le détecteur j du i^e voxel,
- $l_{i,j}$ = la distance parcourue par les rayons X de la j^e projection dans le i^e élément,

$$I_j = I_0 e^{-\sum_i [\mu_i l_{i,j}]} \tag{1.1}$$

$$\Leftrightarrow \frac{I_j}{I_0} = e^{-\sum_i [\mu_i l_{i,j}]}$$

$$\Leftrightarrow \ln\left(\frac{I_j}{I_0}\right) = \ln(e^{-\sum_i [\mu_i l_{i,j}]})$$

$$\Leftrightarrow \ln\left(\frac{I_j}{I_0}\right) = -\sum_i [\mu_i l_{i,j}]$$

$$\Leftrightarrow \ln\left(\frac{I_j}{I_0}\right) = -\langle \mu, l_j \rangle \tag{1.2}$$

L'intensité de départ est choisi lors de l'acquisition, mais peut varier selon la stabilité de la source. L'intensité finale est la mesure que nos détecteurs observent. Les μ_i composent notre image et sont donc les inconnus de notre système. Il ne nous manque que la distance parcourue par chacune de nos projections dans chacun des éléments de notre image. Dû à la nature de la propagation des rayons X, elle peut être approximée. Nous aurons donc besoin d'un outil indispensable pour la reconstruction itérative d'images pour la TDM : la matrice système.

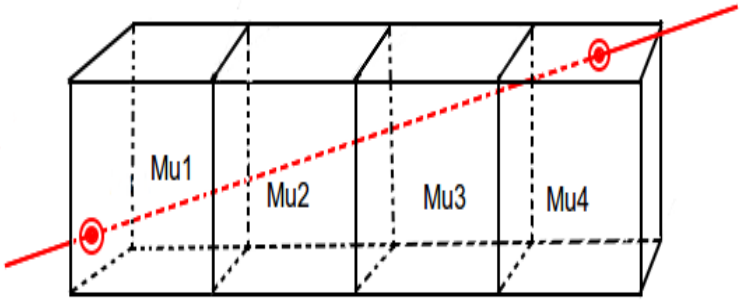


Figure 1.3 – Illustration d'un rayon en TDM, où les différents Mu sont des coefficients d'absortion et où la longueur parcourue dans chaque voxel peut être remplacée par l_{ij}

Les rayons X se propagent majoritairement en ligne droite dans un corps vivant. Ils peuvent dévier, par effet Compton, mais cet effet est négligé ici. Donc, le flux de rayons X partant de la source et allant à un détecteur forme une ligne droite à travers le sujet dans notre cas, ce que nous appelerons une projection. Ainsi, la matrice système de la TDM, dénotée A, est une matrice contenant, pour chaque élément $A_{j,i}$, la distance

parcourue par la j^e projection dans le i^e pixel/voxel. Chaque ligne de A correspond ainsi à une projection différente.

1.2.1 Modèle des moindres carrés

Soit X l'image en deux dimensions de nos μ, donc représentable par une matrice, alors si l'on met bout à bout les colonnes de l'image X et que l'on renomme cette dernière x, alors le produit scalaire entre A et x nous donnera $\langle \mu_j, l_j \rangle$. Si on connaissait la vraie solution X^*, alors on s'attendrait à ce que $Ax = b$, où b représente le terme de gauche en (1.2). On peut donc modéliser notre problème de reconstruction comme un problème de moindres carrés :

$$\arg\min_x ||Ax - b||_2^2, \tag{1.3}$$

avec A la matrice système, x notre image à reconstruire et b, le sinogramme, un vecteur contenant le log du rapport entre l'intensité finale et l'intensité initiale de nos rayons pour chaque projection.

Il est facile d'imaginer l'immensité de la matrice système, qui contient le nombre de pixel au carré de colonnes et au moins autant de lignes. Pour une image de 128 par 128 pixels, cela représente au moins une matrice 16384 par 16384. Toutefois, la plupart des éléments de la matrice système sont nuls. Par exemple, pour une image cubique $N \times N \times N$, il y aura N^3 voxels, mais un maximum de $3N$ intersections. Il est donc très avantageux d'utiliser les structures informatiques dites « Sparse », que l'on traduit par les matrices creuses. Il est tout de même excessivement cher de calculer Ax, ce qui affecte directement le coût du calcul d'un gradient ou d'un hessien du moindre carré. Cette problématique sera abordée plus loin lorsque nous déciderons de nous concentrer sur les algorithmes du

premier ordre.

Le modèle des moindres carrés suppose un bruit gaussien. Or, on sait par le théorème limite central qu'une distribution de Poisson peut-être approximée par une loi Normale lorsque le paramêtre λ de la loi de poisson est assez grand. Même si ce modèle ne respecte pas complètement la physique du problème, son utilisation en pratique fonctionne ([JB18]).

1.2.2 Modèle de la Log-Vraisemblance

Le deuxième modèle que nous présentons utilise le fait que le nombre de photons compté dans chacun des détecteurs suit une loi de Poisson. Pour se faire, on utilise la vraisemblance, qui représente la probabilité qu'une image choisie corresponde aux données observées. Donc, on cherche l'image qui maximise cette métrique. Soit $g(X|I)$ la fonction de vraisemblance de I sachant X , alors on a :

$$\ln(g(X|I)) = \ln(\Pi_j f(I_j|X))$$

$$= \sum_j \ln(f(I_j|X))$$

$$= \sum_j \ln\left(\frac{e^{-\lambda}\lambda^{I_j}}{I_j!}\right)_{\lambda=E[I_j]}$$

$$= \sum_j \ln\left(\frac{e^{-E[I_j]}E[I_j]^{I_j}}{I_j!}\right)_{E[I_j]=I_{0j}e^{-\langle \mu_j, l_j\rangle}}$$

$$= \sum_j [-I_{0j} e^{-\langle \mu_j, l_j \rangle} + I_j \ln(I_{0j}) - I_j \langle \mu_j, l_j \rangle - \ln(I_j!)]$$

$$= \sum_j [-I_{0j} e^{-\langle \mu_j, l_j \rangle} - I_j \langle \mu_j, l_j \rangle + \mathbf{K}]$$

Ici, $\mathbf{K}$ contient les termes qui sont constants par rapport à notre fonction à optimiser. On peut ainsi ne pas en tenir compte, et l'on se retrouve alors avec le problème :

$$\arg\max_{\mu} \sum_j [-I_{0j} e^{-\langle \mu_j, l_j \rangle} - I_j \langle \mu_j, l_j \rangle] \tag{1.4}$$

1.2.3 Variation Totale

On sait que l'intérieur du corps est composé de structures continues de plus ou moins grandes tailles. On peut donc utiliser cette information anatomique comme a priori pour augmenter la robustesse de notre modèle. En effet, si l'on pense à un organe ou à un os par exemple, ce sont des zones homogènes dans ce qui les compose. Localement, les pixels/voxels ne devraient pas varier d'intensité de manière exagérée à l'intérieur d'une de ces structures. Dans le cas contraire, on est sans doute en présence de bruit dans notre image. Comme il a été introduit [ROF92], une manière intéressante d'augmenter la justesse de nos images est de rajouter un terme de régularisation, la variation totale ([HJ11]).

Dans le cas discrétisé, la variation totale pour un pixel isotropique s'exprime comme étant :

$$TV(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sqrt{(x_{i,j} - x_{i+1,j})^2 + (x_{i,j} - x_{i,j+1})^2} + \sum_{i=1}^{m-1} |x_{i,n} - x_{i+1,n}| + \sum_{j=1}^{n-1} |x_{m,j} - x_{m,j+1}|$$

Ce terme est en fait une pénalité que l'on va ajouter à notre fonction objectif, qui punit les changements d'intensité d'un pixel/voxel à l'autre. Il est simple de de mettre les lignes de notre image bouts à bouts, transformation qui changera notre image X pour sa forme vectorisée x. Lorsque combiné à l'équation (1.3), on obtient le problème suivant :

$$\arg\min_{x} ||Ax - b||_2^2 + \lambda ||x||_{TV}. \tag{1.5}$$

Le scalaire λ sert donc de compromis entre notre adéquation aux données et notre a priori de la structure interne du corps. Il faut donc faire attention à l'étendue de ce paramètre, car une valeur trop basse ne traite pas le bruit, mais une valeur trop haute aura tendance à ajouter du flou dans l'image.

Il serait bien de pouvoir profiter de cet a priori anatomique pour le modèle de la Log-Vraisemblance. Peut-on naïvement ajouter un terme de régularisation à notre autre fonction objectif ? On peut le faire en utilisant le modèle du maximum a posteriori. En effet, si g est une distribution a priori sur les paramètres, alors le théorème de Bayes nous donne que :

$$\max_{X} f(X|I) \tag{1.6}$$

$$\Leftrightarrow \max_{X} f(I|X)g(X)$$

Ce qui est le produit de la vraisemblance avec notre fonction a priori. Comme dans l'article [LTK$^+$11], on suppose que $g(x)$ est proportionnelle à $e^{-\lambda TV(X)}$. (Dans cet article, β est l'équivalent de notre λ et $TV(X)$ est l'évaluation de la variation totale pour l'image X). Appliquer le logarithme naturel nous donnera finalement la forme de la vraisemblance logarithmique avec $TV(X)$:

$$\arg\max_{\mu} \sum_{j} [-I_{0j} e^{-\langle \mu_j, l_j \rangle} - I_j \langle \mu_j, l_j \rangle] + \lambda ||X||_{TV}. \tag{1.7}$$

1.3 Tomographie par émission de positons (TEP)

Généralement, en TEP, les patients se font injecter un produit radioactif dans le corps, lequel générera des positons qui s'annihileront avec des électrons. Ces annihilations produisent alors deux photons, qui vont dans des directions anti-parallèles. Lorsque deux photons sont détectés en coïncidence par notre scanner, l'annihilation devrait, à bruit près, s'être produite au long de la droite reliant les détecteurs. On peut alors la trouver. Le nombre de comptes pour un pixel/voxel nous permet alors de donner la valeur de la concentration du produit radioactif pour ce pixel/voxel respectif. La TEP produit une carte de la concentration de notre traceur dans le corps, ce qui en fait une technique d'imagerie médicale fonctionnelle (moléculaire).

Si l'on connaît le comportement de notre traceur lorsqu'il est introduit dans l'objet d'intérêt (le corps par exemple), cette technique devient alors très utile. Un exemple typique est lorsqu'on utilise un traceur basé sur le glucose. Le traceur se concentre alors dans les zones énergivores (telles que les masses cancérigènes) et la vessie.

Le scanner est différent, on ne contrôle plus les radiations comme dans le cas de la TDM, où l'on n'avait besoin que d'une source qui pointe dans une direction et d'un panneau de détecteur placé dans cette même direction. Ici, on n'a aucune idée de la direction dans lequel l'annihilation enverra les deux photons, car c'est uniformément distribué dans une sphère. On a donc avantage à avoir beaucoup plus de détecteurs, couvrant la plus grande superficie possible.

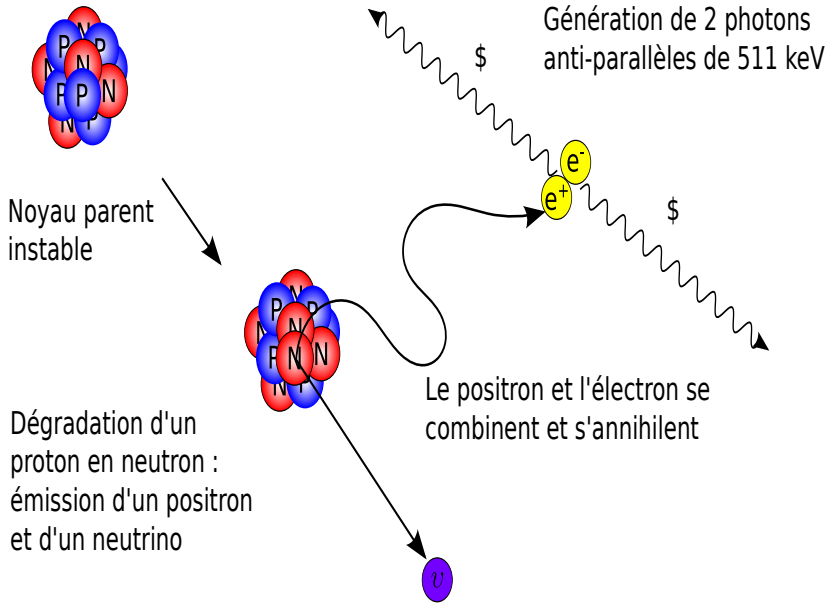


Figure 1.4 – Principe physique derrière la TEP

Comme dans le cas de la TDM, nous aurons une matrice système, mais celle-ci est légèrement différente. En effet, comme ce qui nous intéresse est en fait la concentration dans chaque pixel/voxel et non l'atténuation de l'intensité le long d'une droite, alors la relation entre l'image recherchée et les données n'est pas définie à l'aide de longueur d'intersection. La matrice système en TEP contient autant de lignes qu'il y a de paire de détecteurs plausible. En effet, considérant la structure de l'appareil, la droite reliant certaines paires de détecteurs ne passent même pas à l'intérieur de l'objet. Les éléments de la ligne i sont alors : la probabilité pour que, si un compte est émis du pixel/voxel j, il soit détecté dans la projection i. Cette matrice est toute aussi immense que la matrice système en TDM. Nous avons donc les mêmes restrictions quand à la mémoire et aux dérivées d'ordre supérieur à un. Il existe plusieurs phénomènes qui influencent l'acquisition en TEP et qui peuvent être incorporés dans la matrice système, mais ils n'ont pas été considérés dans nos travaux. Ainsi, la matrice utilisée pour nos expérimentations est une version

maison simplifiée.

1.3.1 Modèle de la Log-Vraisemblance

Pour une paire de détecteurs j, on sait que le nombre de comptes suivra une loi de Poisson de paramètres $\lambda = \sum_i A_{ji}x_i$. Avec A_{ji} un élément de la matrice système, et x_i le pixel/voxel correspondant. Cela est dû au fait que pour chaque pixel/voxel x_i, l'espérance du nombre d'émission de positons qui produisent des paires qui s'annihilent dans la direction de la projection j suit une loi de Poisson de paramètre $A_{ji}x_i$. Le nombre de comptes pour une paire de détecteurs est donc la somme du nombre de compte, et ce pour chaque pixel/voxel se trouvant en possibilité d'envoyer une paire de photons sur les deux détecteurs formant ladite paire. Or, la somme de loi de Poisson indépendante est une loi de Poisson dont le paramètre est la somme des paramètres des dites lois de Poisson. De ce fait, nous pouvons donc définir la vraisemblance de notre problème comme étant :

$$p(y|x) = \prod_j \frac{e^{(-\sum a_{ji}x_i)}(\sum a_{ji}x_i)^{y_j}}{y_j!}.$$

Le problème de reconstruction peut être écrit sous la forme d'un problème de maximisation de la Log-Vraisemblance :

$$\arg\max \ln(p(y|x)) = \sum_j [-\sum a_{ji}x_i + y_j \ln(\sum a_{ji}x_i)]$$

$$\iff \arg\max \ln(p(y|x)) = \sum_j [-Ax + y_j \ln(Ax)] \tag{1.8}$$

On verra dans le deuxième chapitre que cette formulation admet une solution analytique.

1.3.2 Modèle des moindres carrés

Comme pour le cas de la TDM, il est possible de résoudre le problème de reconstruction en TEP en utilisant les moindres carrés. Même si ce modèle diverge un peu au niveau théorique, en pratique, il a déjà été utilisé de manière fructueuse par le passé ([Kau93]). On a que le produit scalaire Ax devrait correspondre à espérance près au nombre de comptes pour une paire de détecteurs donnée, et les données issues du TEP-Scan sont en fait le nombre de comptes enregistrés durant l'acquisition. On peut donc tenter de trouver l'image x qui minimise :

$$\arg\min_x ||Ax - b||_2^2 \tag{1.9}$$

1.3.3 Variation Totale

L'utilisation de la variation totale est encore une fois justifiable, puisque l'a priori que l'on avait fonctionne encore. En effet, si certaines structures sont des points d'attraction d'un radiotraceur injecté, celle-ci devrait varier peu localement. Les pixels/voxels représentant une même structure auront tendance à représenter une continuité locale dans leur concentration. On peut donc ajouter un terme de variation totale aux équations (1.8) et (1.9) comme dans (1.5) et ainsi obtenir nos modèles régularisés :

$$\arg\max_x \sum_j [-Ax + y_j \ln(Ax)] - \lambda ||X||_{TV}, \tag{1.10}$$

$$\arg\min_x ||Ax - b||_2^2 + \lambda ||X||_{TV}. \tag{1.11}$$

1.4 Nature du bruit

Ces deux modalités d'imagerie admettent un bruit poissonnien issus du comptage de photons, eux-mêmes suivants des exponentielles comme vu en (1.1). Leur bruit suit donc une loi de Poisson, ayant pour paramètre le nombre de comptes. Cependant, les rayons X sont en très grands nombres par projection, et le théorème de la limite centrale nous donne qu'une loi de Poisson de paramètre λ s'approxime très bien par une loi Normale lorsque λ est grand (Notamment, plus de 30 est une bonne approximation, plus de mille est une très bonne approximation). Ainsi, il n'est pas rare d'utiliser un bruit gaussien pour simuler le bruit relié à l'acquisition en TDM. Il existe d'autres sources de bruit pouvant affecter nos images, tel que vu dans [SV82], mais pour les fins de notre étude, nous nous somme limité à celui qui était le plus fondamental. En effet, les autres bruits entraînaient des choix trop spécifiques quant à la méthodologie et à la résolution. Comme le but du mémoire est de comparer le potentiel de différents algorithmes, nous avons choisi d'omettre ces bruits supplémentaires.

1.5 Contraintes de non-négativité

Pour nos deux modalités, on a que les variables sont bornées inférieurement par zéro. En effet, en TEP, on n'aura jamais de concentration négative, cela ne fait pas de sens au niveau physique. On a donc avantage à utiliser des algorithmes de reconstruction capables de gérer les contraintes de borne, ou bien utiliser des méthodes capables de modifier nos algorithmes afin qu'ils les gèrent. La même contrainte s'applique au cas de la TDM. Une manière de voir cet argument est qu'une valeur négative dans notre image correspondrait à une augmentation de l'intensité de notre rayon sur la distance parcourue dans ce voxel, alors que le corps humain ne contient pas de source ionisante.

Une autre manière de voir cet argument est que si l'on reprend l'équation (1.2), et que l'on isole le produit scalaire, on obtient :

$$\ln\left(\frac{E[I]}{I_0}\right) = -\langle \mu_j, l_j \rangle$$

$$\Longleftrightarrow$$

$$\ln(I_0) - \ln(E[I]) = \langle \mu_j, l_j \rangle$$

Or, on sait que I_0 est plus grand que $E[I]$, de même pour leur logarithme respectif. Le produit scalaire est donc toujours supérieur à zéro, alors qu'une des composantes de ce produit scalaire est une distance. La non-négativité des distances donnerait donc que pour toutes les projections possibles, le produit scalaire soit positif, ce qui ne se fera que si les éléments sont tous positifs ou égaux à zéro.

Comme la gestion des contraintes est différente selon l'algorithme de reconstruction utilisé et le modèle, la discussion sera faite dans la section méthodologie, après avoir introduit chacun des algorithmes.

1.6 Les problèmes que l'on peut rencontrer

1.6.1 Manque de données

Une méthode pour augmenter la fiabilité des images serait d'utiliser plus d'informations lors de l'acquisition. Cependant, dans les cas étudiés, plus d'informations implique aussi plus de radiations pour le patient, ce qui est rarement une option. On cherche donc les

algorithmes de reconstruction tomographique qui fonctionneront le mieux sur de faibles jeux de données. On cherche ainsi à maximiser la justesse de notre image reconstruite, tout en minimisant les expositions nocives pour les patients.

1.6.2 Tailles des matrices systèmes

Les matrices systèmes pour nos deux types de problèmes (TEP et TDM) sont de gigantesques conteneurs d'information, qui nous informe des milieux parcourus par nos projections. Si nous voulions utiliser les matrices hessiennes, nous aurions besoin de calculer la matrice A^tA, une matrice carrée, symétrique et non creuse dont les dimensions sont égales au nombre de pixel/voxel de notre image. Cette dimensionnalité devient rapidement contraignante dans des cas d'application réelle. Il est cependant possible d'utiliser des algorithmes qui calculent des produits hessien-vecteurs. Cette astuce calculatoire permet d'obtenir le résultat d'opérations telle que A^tAx en calculant à la place $A^t(Ax)$. On arrive ainsi à ne pas reconstituer la matrice hessienne, tout en utilisant des algorithmes plus complexes. Ces méthodes ne sont pas dans le mémoire, mais elles seraient un bel ajout pour un futur travail.

CHAPITRE 2

Algorithmes du premier ordre

Ce chapitre est un résumé des différents algorithmes que nous abordons dans cet ouvrage. Nous avons décidé d'envisager plusieurs familles différentes d'algorithmes. Cette liste n'est pas complète, mais nous avions facilement accès à ceux-ci et la compétitivité de la plupart d'entre eux est réputée.

2.1 Pourquoi visons-nous le premier ordre ?

Comme la reconstruction de la matrice hessienne est trop lourde en terme de temps de calcul, nous visons les algorithmes d'ordre un, ce que nous définissons par les algorithmes n'utilisant qu'au plus des fonctions objectifs et des gradients. Cependant, nous utiliserons des algorithmes capables d'utiliser de l'information du deuxième ordre, et ce, en profitant de la méthode sécante.

2.2 Définitions liées à la convergence

Tout au long de l'ouvrage, il sera fait mention de concepts tels que la convergence globale, la vitesse de convergence globale et la vitesse de convergence asymptotique. Nous jugions qu'il serait donc important de les définir au préalable.

On dit d'un algorithme qu'il est globalement convergent s'il converge vers un point stationnaire, et ce peu importe le point de départ x_0.

La vitesse de convergence globale est la quantité k de ressources requises pour atteindre une précision donnée ϵ. On peut voir k comme étant un nombre d'itération ou encore, le nombre d'appel à la fonction objectif ou à l'évaluation du gradient ([Nes83b]).

Lorsque nous sommes dans le cas convexe, alors ϵ est environ $1/k$, avec un optimal à $1/k^2$. Lorsque fortement convexe, ϵ suit un certain taux τ_1^k, où $\tau_1 = (1-r)/(1+r)$ dans le cas général, et dans le cas optimal, τ_2^k, où $\tau_2 = (1-\sqrt{r})/(1+\sqrt{r})$. Nous en reparlerons à la prochaine section.

La vitesse de convergence asymptotique est définie lorsque l'objectif est fortement convexe. Elle s'exprime comme étant :

$$\limsup_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^q} = \alpha < \infty.$$

L'ordre de convergence asymptotique sera alors la plus grande valeur de q telle que cette inégalité demeure respectée.

Cette dernière est inspirée de la section 1.4 du mémoire de Catherine Simard [Sim15]. Pour des informations plus approfondies et très bien vulgarisées, je vous conseille cet ouvrage.

2.3 Convexité

Nous avions mentionné plus tôt que notre objectif était convexe, mais nous voulions savoir si nous avions aussi la forte convexité. Nous avons ainsi regardé l'évolution de la norme infinie du gradient de notre problème.

Dans le cas fortement convexe, une convergence linéaire de taux $\frac{1-r}{1+r}$ est atteinte par les algorithmes de type gradients, alors que les algorithmes optimaux atteignent une convergence linéaire de taux $\frac{1-\sqrt{r}}{1+\sqrt{r}}$, où r est le ratio de la constante de forte monotonie divisé par la constante de Lipschitz de notre gradient. Ainsi, si l'objectif a une constante de forte convexité qui tend vers 0, nous obtiendrons une convergence sous-linéaire.

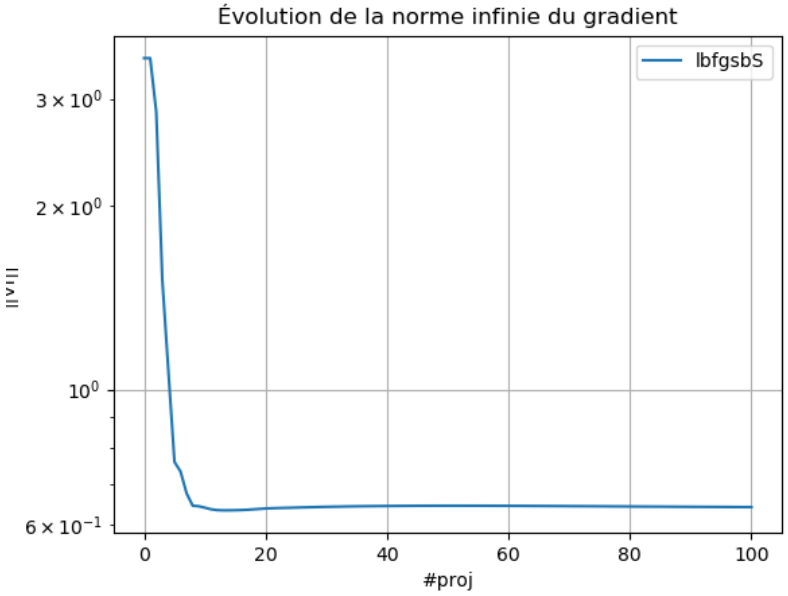


Figure 2.1 – Évolution de la norme infinie du gradient - L'évolution de la norme infinie du gradient de notre modèle utilisant la vraisemblance logarithmique laisse supposer que l'objectif n'est pas fortement convexe, ou que sa constante de forte convexité est très faible

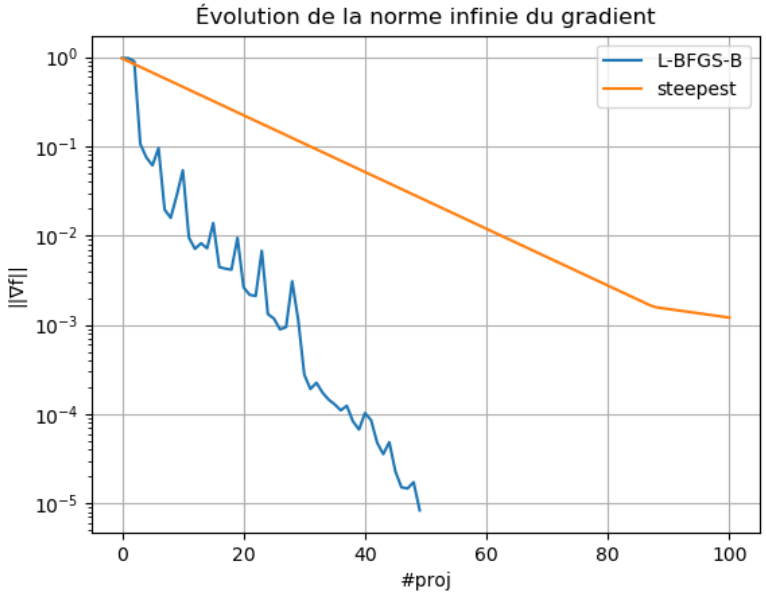


Figure 2.2 – L'évolution de la norme infinie du gradient d'une fonction strictement convexe

Dans la figure 2.1 en regardant l'évolution de la norme de notre gradient par notre nombre d'itérés, on remarque que ça s'apparente à une forme $\frac{1}{k}$, avec k le nombre d'itérés. On aurait ainsi une expérimentation montrant une convergence sous-linéaire globale telle qu'étudiée par Nesterov. Dans ce cas, on s'attend à ce que FISTA affiche de meilleures performances que les autres algorithmes car il est conçu pour avoir une vitesse globale de convergence optimale de $1/k^2$

À titre de comparaison, en 2.2, le même graphique a été généré avec une fonction strictement convexe : un paraboloïde de dimension 100 avec des coefficients aléatoires utilisant la graine aléatoire 1234, et les algorithmes L-BFGS-B (utilisé sans contraintes de bornes) ainsi qu'une descente de gradient (Steepest descent).

2.4 Descente de gradient

2.4.1 Origine

La descente de gradient remonte à Cauchy en 1847 [Cau47]. Cet algorithme est très simple. Sachant que le gradient représente la direction de la pente la plus forte, nous avons que l'inverse du gradient est une direction de descente. Ainsi, allez du point courant vers un nouveau point se trouvant sur un certain multiple de l'inverse de notre gradient fera diminuer la valeur de notre fonction objectif. Nous pouvons ainsi utiliser une recherche linéaire $h_{x,d}(\theta)$ afin de trouver la longueur du pas que nous prenons dans la direction de l'inverse du gradient. Lorsque notre fonction objectif est convexe et que son gradient est Lipschitz-continu avec une constante Lipschitz L, un pas fixe de $1/L$ assure une convergence. Comme la vraisemblance logarithmique et les moindres carrés sont tous deux des problèmes convexes, ces méthodes sont à tester. Pour nos tests, nous utilisons des algorithmes avec pas fixes (FISTA par exemple), et d'autres utilisant des recherches linéaires (L-BFGS-B, gradients conjugués non-linéaires, ...). On peut ainsi boucler jusqu'à l'obtention d'un minimum local x^*. Il est à noter que pour un problème de maximisation, on nomme la méthode : ascension du gradient.

2.4.2 Pseudo-code

Algorithme 1 : Descente de gradient

Données : $x \in \mathbb{R}^n$
Résultat : $x^* \in \mathbb{R}^n$
1 **tant que** $\neg$ *convergence* || *épuisé* **faire**
2 $\quad d = -\nabla f(x)$;
3 $\quad \theta^* = \arg\min_{\theta} h_{x,d}(\theta)$;
4 $\quad x^+ = x + \theta^* d$;
5 $\quad x = x^+$;

2.4.3 Ordre de convergence

Selon Nesterov [Nes14], les descentes de gradient offrent un taux de convergence global linéaire pour un problème fortement convexe, et un taux asymptotique linéaire. Nesterov mentionne aussi que ce taux global bat en général celui des méthodes quasi-Newtonienne et celui des méthodes de gradients conjugués. Cependant, asymptotiquement, les descentes de gradients sont très faibles par rapport à celles-ci.

Dans notre cas convexe, ou bien fortement convexe avec une constante de forte convexité extrêmement petite, la convergence sera de l'ordre de $1/k$, soit sous-linéaire.

2.4.4 Avantages/désavantages

Bien que cette méthode soit extrêmement simple à implanter et très peu coûteuse en terme de calculs, elle est cependant très lente ([Dus17]).

2.5 MLEM

2.5.1 Origine

La méthode itérative MLEM (Maximum Likelihood Expectation Maximisation) est une technique qui permet de générer un algorithme itératif pour trouver le maximum d'une fonction de vraisemblance. Si l'on transforme le problème (1.10) en problème de minimisation, on obtient :

$$\arg\min_x \sum_j [Ax - y_j \ln(Ax)] + \lambda ||X||_{TV}. \tag{2.1}$$

Or, on ne peut pas trouver une solution analytique sous cette forme. Il serait cependant aisé de résoudre ce problème d'optimisation si l'on avait les informations des concentrations pour chaque pixel/voxel composant notre tube j ayant pour total de coïncidences y_j. Nous allons donc remplacer nos totaux y_j par une somme sur des nouvelles variables $z_{j,i}$, qui représenteront le nombre de photons enregistrés dans le tube j et émis par le pixel/voxel i. Dans ce cas, le logarithme de l'intégrale de ligne devient désormais le logarithme d'un scalaire. Notre modèle devient alors :

$$\mathcal{L}(z,x) = \sum_j \left[\sum_i a_{j,i}x_i - z_{j,i} \ln(a_{j,i}x_i) \right] - \lambda ||X||_{TV}, \tag{2.2}$$

où l'on décrit la vraisemblance logarithmique par $\mathcal{L}(z,x)$.

Comme on ne connaît pas les valeurs de z_i, il faut les approximer. On sait que nos $z_{j,i}$ suivent une loi de Poisson de paramètre $a_{j,i}x_i^*$. En utilisant l'espérance conditionnée par notre estimé courant et notre sinogramme, nous pourrons approximer les valeurs des z_i. Or, l'espérance lorsque l'on conditionne une variable aléatoire issue d'une loi de Poisson par la somme de variables aléatoires issues de lois de Poisson indépendantes (incluant celle que l'on conditionne) revient à :

$$\mathbb{E}(z_{j,i} | \Sigma_i z_{j,i} = y_j, x_i^{k-1}) = \frac{y_j a_{j,i} x_i^{k-1}}{\sum_i a_{j,i} x_i^{k-1}}.$$

Cela vient du fait que l'on retrouve l'espérance d'une loi Multinomiale [HL89]. Le sous-problème que l'on obtient est donc :

$$\mathbb{E}\left[\mathcal{L}(z,x) | x^{k-1}, y\right] = \sum_j \sum_i a_{j,i}x_i - \sum_j \sum_i \frac{y_j a_{j,i} x_i^{k-1}}{\sum_q a_{j,q} x_q^{k-1}} \ln(a_{j,i}x_i) - \lambda ||X||_{TV}.$$

Or, comme l'ont démontré Shepp et Vardi dans [SV82], la fonction objectif est convexe,

donc trouver un point stationnaire nous assure de trouver un minimum. On trouve ainsi que la dérivée de notre espérance s'annule pour :

$$x^k = \frac{1}{\sum_j a_{j,i} - \beta \nabla TV(x)} \sum_j \frac{y_j x_i^{k-1} a_{j,i}}{\sum_q a_{j,q} x_q^{k-1}}$$

2.5.2 Pseudo-code

Algorithme 2 : MLEMTV

Données : $x_0 \in \mathbb{R}^n$

Résultat : $x^* \in \mathbb{R}^n$

1 **tant que** $\neg$ *convergence* & *épuisé* **faire**

2 $\quad x^k = \frac{1}{\sum_j a_{j,i} - \beta \nabla TV(x^{k-1})} \sum_j \frac{y_j x_i^{k-1} a_{j,i}}{\sum_q a_{j,q} x_q^{k-1}}$;

2.5.3 Ordre de convergence

Les descentes de gradient offrent un taux de convergence global linéaire dans un problème fortement convexe, et un taux asymptotique linéaire selon Nesterov [Nes14]. Il mentionne aussi que ce taux global bat en général celui des méthodes quasi-Newtonienne et celui des méthodes de gradients conjugués. Cependant, asymptotiquement, les descentes de gradients sont très faibles par rapport à celles-ci. Comme il sera expliqué à la dernière sous-section, MLEM est l'équivalent d'une ascension du gradient, ils partagent donc un ordre de convergence semblable.

2.5.4 Avantages/désavantages

MLEM est reconnue pour reconstruire le bruit à partir d'un certain moment critique [Kad01]. L'algorithme doit donc être arrêté à ce moment, afin d'éviter la reconstruction du bruit. Utiliser une régularisation dans MLEM exploitant l'a priori permet de mitiger

le sur-ajustement (overfitting). Cette affirmation sera confirmée dans le premier test du chapitre cinq. De plus, la version multiplicative de l'algorithme gère naturellement les contraintes de non-négativité en ne multipliant l'image que par des éléments positifs, à condition que l'image de départ est elle aussi strictement non-négative.

2.5.5 Représentation sous la forme d'une descente de gradient

Lorsque nous récrivons le MLEM sous une forme additive, on voit que cette méthode est équivalente à une ascension du gradient, sauf qu'elle est pondérée de manière à garder le nombre de comptes dans l'image intacte au fil des itérations.

Soit $\otimes$ l'opérateur de multiplication point à point et $\oslash$ son équivalent pour la division, la formule du MLEM multiplicatif :

$$X_{k+1} = (X_k \otimes C) \oslash R, \tag{2.3}$$

tel que $C = A^t(b \oslash (AX_k))$ et $R = A^t\mathbb{1} + \beta TV(X_k)$,

alors MLEM équivaut à une ascension du gradient pondérée.

Démonstration. On note ici qu'en posant $\beta = 0$, on obtiendra le MLEM non régularisé, et donc que cette preuve couvre les deux versions de MLEM que nous utilisons.

On ajoute et retranche X_k, puis on met sur le même dénominateur :

$$X_{k+1} = X_k + (X_k \otimes C) \oslash R - (X_k \otimes R) \oslash R \tag{2.4}$$

On met X_k en évidence :

$$X_{k+1} = X_k + (X_k \otimes (C - R)) \oslash R \tag{2.5}$$

Ce qui revient à :

$$X_{k+1} = X_k + (X_k \oslash R) \otimes (C - R) \tag{2.6}$$

Or, $C - R$ revient en fait à l'expression de notre gradient, et on reconnaît donc ici une ascension du gradient, sauf qu'elle est pondérée.

□

2.6 L-BFGS

L'explication de L-BFGS et de L-BFGS-B passe naturellement par une explication détaillée de l'algorithme BFGS. Voici donc les grandes étapes fondatrices de BFGS, puis les étapes supplémentaires menant au L-BFGS et puis au L-BFGS-B.

2.6.1 Origine

BFGS est l'algorithme de type quasi-Newton le plus populaire, dont le nom est issus des initiales des chercheurs l'ayant découvert en même temps de manière indépendante : Broyden, Fletcher, Goldfarb et Shanno[[Bro70], [Fle70], [Gol70] et [Sha70]]. Dans les méthodes quasi-Newtonienne, l'information du premier ordre est utilisée afin d'approximer la matrice Hessienne. Ceci permet de profiter d'une partie des avantages de l'information du deuxième ordre, et ce, sans avoir à calculer la matrice Hessienne, qui est trop lourde à calculer à chaque itération.

Lorsqu'on utilise la direction de Newton, on veut minimiser l'approximation quadratique de f pour un déplacement p. On obtient ainsi :

$$m_k(p) = f_k + \nabla f_k^t p + \frac{1}{2} p^t \nabla^2 f_k p,$$

qui admet comme minimiseur :

$$p_k = -\nabla^2 f_k^{-1} \nabla f_k.$$

Si le même raisonnement est repris, mais en utilisant une approximation B_k de la Hessienne $\nabla^2 f_k$, nous obtenons :

$$m_k(p) = f_k + \nabla f_k^t p + \frac{1}{2} p^t B_k p,$$

qui admet comme minimiseur :

$$p_k = -B_k^{-1} \nabla f_k.$$

Ce minimiseur nous sert ensuite de direction de descente, et en utilisant α_k issus d'une recherche linéaire :

$$x_{k+1} = x_k + \alpha_k p_k.$$

Si on construit une approximation quadratique au nouveau point x_{k+1}, on obtient :

$$m_{k+1}(p) = f_{k+1} + \nabla f_{k+1}^t p + \frac{1}{2} p^t B_{k+1} p.$$

On veut que le précédent et le nouveau gradient corresponde avec ∇m_{k+1}. Or, pour le nouveau gradient, il est clair que $\nabla m_{k+1}(0) = \nabla f_{k+1}^t$. Pour l'ancien, on a que :

$$\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1}^t + \alpha_k B_{k+1} p_k,$$

$$= \nabla f_k \iff B_{k+1} s_k = y_k.$$

On appelle cette dernière égalité la condition sécante. Nous aimerions aussi que cette matrice soit définie positive. Si l'on multiplie la condition sécante à gauche par s_k^t, on obtient :

$$s_k^t B_{k+1} s_k = s_k^t y_k,$$

qui se réécrit, en considérant que B_{k+1} est définie positive, par :

$$s_k^t y_k > 0.$$

On appelle cette dernière la condition de courbure.

Afin d'assurer l'unicité de B_{k+1}, on la voudra la plus près possible de la matrice B_k. On résoudra donc le problème :

$$\arg\min_B ||B - B_k||,$$

sujet à :

$$B = B^t, Bs_k = y_k.$$

En choisissant une norme comme dans [NW06], on obtient que la solution de ce problème est :

$$B_{k+1} = (\mathbb{1} - \rho_k y_k s_k^t) B_k (\mathbb{1} - \rho_k s_k y_k^t) + \rho_k y_k y_k^t,$$

tel que $\rho_k = \frac{1}{y_k^t s_k}$ et $\mathbb{1}$ est un vecteur de 1.

Il est cependant beaucoup plus efficace de passer par le dual de ce problème, et de recréer directement l'inverse de cette approximation de la Hessienne. On obtient alors que la mise à jour BFGS est :

$$H_{k+1} = (\mathbb{1} - \rho_k s_k y_k^t) H_k (\mathbb{1} - \rho_k y_k s_k^t) + \rho_k s_k s_k^t. \tag{2.7}$$

2.6.2 Pseudo-code

Algorithme 3 : BFGS

Données : x_0 un point de départ et H_0 une première approximation de H
Résultat : x^*

1 $k = 0$;
2 **tant que** ¬ *convergence* & *épuisé* **faire**
3 On calcule la direction $p_k = -H_k \nabla f_k$;
4 On calcule $x_{k+1} = x_k + \alpha_k p_k$, tel que α_k est issus d'une recherche linéaire;
5 On calcule $H_{k+1} = (\mathbb{1} - \rho_k s_k y_k^t) H_k (\mathbb{1} - \rho_k y_k s_k^t) + \rho_k s_k s_k^t$;
6 k = k+1 ;

L'algorithme L-BFGS, (limited memory-BFGS), est une version allégée qui permet de traiter des problèmes de dimension immense, où la matrice Hessienne est tellement grande qu'on ne pourrait même pas la garder en mémoire. Le raisonnement derrière cette modification est que l'on voudrait avoir de l'information localement sur la courbure de notre fonction objectif. On oubliera ainsi de l'information que l'on considère désuète pour faire

place aux informations plus récentes. Ainsi, au lieu de garder une matrice $n \times n$ en mémoire, on gardera m paires de vecteurs de dimension n, avec m entre trois et vingt selon [NW06], qui nous donneront l'équivalent d'une matrice H_{k+1} à l'aide d'une mise à jour de rang deux. L'utilisation successive de la formule (2.7) fournit une procédure récursive (comme vu dans [NW06]) qui nous permet de calculer le produit $H_{k+1}\nabla f_{k+1}$ pour trouver notre direction de descente directement. L'algorithme devient alors :

Algorithme 4 : L-BFGS

Données : x_0 un point de départ et H_0 une première approximation de H
Résultat : x^*

1 $k = 0$;
2 **tant que** ¬ *convergence* & *épuisé* **faire**
3 On choisit un H_k^0 ;
4 On calcule la direction $p_k = -H_k\nabla f_k$, selon la procédure récursive de L-BFGS ;
5 On calcule $x_{k+1} = x_k + \alpha_k p_k$, tel que α_k est issus d'une recherche linéaire;
6 On met la banque de vecteurs à jour avec les nouveaux s_k et y_k ;
7 $k = k + 1$;

On note que si $n < m$, L-BFGS revient exactement à BFGS.

Le dernier ajout qui sera fait servira à la gestion des bornes. [BLNZ95] ont implémenté une version de L-BFGS qui prend en compte les contraintes de bornes supérieures et inférieures, s'appelant L-BFGS-B. Cet algorithme est célèbre dans plusieurs domaines, et son utilisation revient presque toujours à des interfaces entre le langage utilisé et sa version implémentée en FORTRAN77 par les auteurs originaux. La différence majeure avec L-BFGS, c'est que la direction de descente sera trouvée après avoir utilisé un algorithme de projection de gradients (afin de trouver les contraintes actives). Ces contraintes seront bloquées sur leur borne, et la minimisation de l'approximation quadratique se fera sur les variables qui sont encore libres.

2.6.3 Ordre de convergence

Toujours selon le livre de Nesterov [Nes14], et toujours dans le cas d'un objectif fortement convexe, la convergence asymptotique de l'algorithme BFGS est super-linéaire, quoique supérieure théoriquement à celle des méthodes de gradient conjugué. Pour ce qui est des algorithmes BFGS à mémoire limitée, la convergence n'est pas super-linéaire mais elle reste rapide. Pour ce qui est de la convergence globale, théoriquement, elle reste en deçà de celle de la descente de gradient. On s'attend donc à avoir une meilleure convergence si l'on utilise un point de départ meilleur qu'une simple image uniforme. Le quatrième test tentera de vérifier cette supposition.

Dans notre cas convexe, ou bien fortement convexe avec une constante de forte convexité extrêmement petite, la convergence sera d'un ordre se situant entre $1/k$ et $1/k^2$. On sait que pour une fonction quadratique, BFGS obtiendrait $1/k^2$, mais la version à mémoire limitée de cet algorithme n'atteint pas la borne optimal.

2.6.4 Avantages/désavantages

Comme L-BFGS et L-BFGS-B sont conçus pour des problèmes de grandes dimensions, et ayant une bonne convergence, ils semblent tout enclin à offrir de superbes performances pratiques. De plus, la gestion naturelle des bornes par L-BFGS-B rend la technique d'autant plus alléchante. Un désavantage de ces techniques serait cependant les hyper-paramètres m et H_k^0, qui sont issus d'heuristique, et donc difficiles à justifier.

2.7 Gradient conjugué non-linéaire

Le gradient conjugué est un algorithme général, issus de l'optimisation des années 1950-1970. Nous voulons l'inclure dans nos analyses, considérant ses forces et sa simplicité.

2.7.1 Origine

Le gradient conjugué non-linéaire est une adaptation aux fonctions non-quadratiques de la célèbre formule de Hestenes et Stiefel [HS52]. Ainsi, Fletcher et Reeves [FR64] ont proposé une version pour les fonctions non-quadratiques et même non-convexes. L'algorithme construit des directions conjuguées par conjugaison des résidus.

2.7.2 Pseudo-code

Algorithme 5 : Gradient conjugué non-linéaire, version générale

Données : x_0
Résultat : x^*
1 initialisation : $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0)$;
2 *soit* $d_0 = -\nabla f_0$;
3 **tant que** $\neg$ *convergence* & *épuisé* **faire**
4 On trouve α_k;
5 $x_{k+1} = x_k + \alpha_k d_k$;
6 On évalue ∇f_k;
7 $d_{k+1} = -\nabla f_k + \beta_{k+1} d_k$;
8 $k = k + 1$;

Pour cet algorithme, α_k est issus d'une recherche linéaire, et β_k est une formule qui varie d'une version de gradient conjugué à une autre. La notation usuelle veut que y_k soit la différence entre les deux derniers gradients $\nabla f(x_{k+1})$ et $\nabla f(x_k)$. Les différentes versions que nous utilisons sont :

Fletcher et Reeves

$$\beta_k = \frac{\nabla f(x_{k+1})^t \nabla f(x_{k+1})}{\nabla f(x_k)^t \nabla f(x_k)}$$

Polak et Ribière

$$\beta_k = \frac{\nabla f(x_{k+1})^t y_{k+1}}{\nabla f(x_k)^t \nabla f(x_k)}$$

Hestenes et Stiefel

$$\beta_k = \frac{\nabla f(x_{k+1})^t y_{k+1}}{y_{k+1}^t d_k}$$

Hager et Zhang

$$\beta_k = \frac{1}{y_{k+1}^t d_k} \left(y_{k+1}^t - 2 d_k^t \frac{||y_{k+1}||^2}{y_{k+1}^t d_k} \right) g_{k+1}$$

Il est à noter que ces différentes versions sont toutes équivalentes pour des problèmes quadratiques. Il y a d'autres formules, mais on se limite à celles-ci car elles sont plus répandues. Selon leurs tests, c'est la version de Hager et Zhang qui aurait les meilleures performances.

2.7.3 Ordre de convergence

Toujours selon le livre de Nesterov [Nes14], et dans un cas où l'objectif est fortement convexe, la convergence asymptotique de l'algorithme est super-linéaire, quoique inférieure théoriquement à celle des méthodes quasi-Newtonienne. Pour ce qui est de la convergence globale, théoriquement, elle reste en deçà de celle de la descente de gradient. On s'attend donc ici aussi à avoir une meilleure convergence si l'on utilise un point de départ meilleur qu'une simple image uniforme.

Dans notre cas convexe, ou bien fortement convexe avec une constante de forte convexité extrêmement petite, la convergence sera d'un ordre se situant entre $1/k$ et $1/k^2$. On sait que pour une fonction quadratique, on obtiendrait $1/k^2$, mais sinon le gradient conjugué n'atteint pas la borne optimal.

2.7.4 Avantages/désavantages

Ces méthodes sont reconnues comme étant exceptionnellement performantes dans le cas quadratique, convergent en au plus n itérations, lorsque la dimensionnalité du problème est n. Cependant, dans le cas où les gradients sont non-linéaires, (fonction non-quadratique), ces méthodes sont très sensibles aux erreurs de calculs associées aux arrondis. Un mauvais conditionnement entraîne donc des stagnations de l'algorithme, ce qui peut même entraîner une non-convergence. Cependant, l'algorithme se comporte bien lorsqu'on lui ajoute une technique de réinitialisation. De plus, comme on le verra en annexe, un préconditionneur obtenu économiquement de manière judicieuse peut être utilisé afin d'améliorer les performances de l'algorithme. Un des points forts des méthodes de gradients conjugués est son peu de ressources nécessaires en termes de mémoire. Du côté computationnel, pour des problèmes de très grandes tailles sans contraintes, c'est une technique très performante. Comme notre problème comporte des contraintes de bornes,

mais que cette famille d'algorithmes ne les gère pas naturellement, nous parlerons d'une méthode générale de gestion des bornes dans la section méthodologie, qui peut être ajoutée aux différents modèles, ainsi que de notre implémentation d'une version simplifiée qui est encourageante pour justifier une implémentation future.

2.8 FISTA

On explique ici le fonctionnement général de l'algorithme FISTA. Un chapitre ultérieur sera consacré à l'application de FISTA sur le problème de reconstruction tomographique régularisée.

2.8.1 Origine

FISTA (Fast-Iterative Shrinkage Tresholding Algorithm) est une version accélérée d'un algorithme déjà bien connu, ISTA. Soit :

$$\tau_\alpha(x)_i = (|x_i| - \alpha)_+ * sgn(x_i),$$

alors l'étape générale de ISTA, comme expliquée dans [BT09b] est :

$$x_{k+1} = \tau_{\lambda t}(x_k - 2tA^t(Ax_k - b)),$$

où t est issus d'une recherche linéaire approximative.

Si on prend le modèle général suivant :

$$\min F(x) := f(x) + g(x), x \in \mathbb{R}^n, \tag{2.8}$$

avec $f(x)$ une fonction convexe de $\mathbb{R}^n \to \mathbb{R}$ de type C^1, et $g(x)$ une fonction continue, convexe et possiblement non-différentiable.

Utilisons une approximation quadratique du problème 2.8 :

$$Q_L(x, y) = f(y) + \langle (x - y), \nabla f(y) \rangle + \frac{L}{2}||x - y||^2 + g(x),$$

alors l'étape générale de ISTA devient :

$$x_k = p_L(x_{x-1}), \tag{2.9}$$

avec p_L l'opérateur proximal :

$$p_L(y) = \arg\min x \Big\{ g(x) + \frac{L}{2} \Big\| x - \Big(y - \frac{1}{L} \nabla f(y) \Big) \Big\|^2 \Big\}.$$

On obtient ici la version proximale de ISTA, qui nous sera utile lors de la présentation de FISTA. En effet, les auteurs de [BT09b] arrivent à étendre ISTA à l'algorithme de Nesterov datant de 1983 [Nes83a], un algorithme étant conçu pour atteindre la borne de pire cas au sens de Nemirovsky et Yudin dans [NY83]. Le résultat de cette adaptation s'appelle FISTA.

2.8.2 Pseudo-code

Algorithme 6 : FISTA

Données : L la constante de Lipschitz de ∇f, x_0 un point de départ dans $\mathbb{R}^n$
Résultat : x^*

initialisation :$y_1 = x_0$ et $t_1 = 1$;
tant que $\neg$ *convergence* & *épuisé* **faire**
  $x_k = p_L(y_k)$;
  $t_{k+1} = \frac{1+\sqrt{1+4t_k}}{2}$;
  $y_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1})$;

L'algorithme résultant est donc très semblable à ISTA, mais il utilise l'opérateur proximal sur y_k, qui est une combinaison linéaire judicieusement choisie des deux derniers points visités.

2.8.3 Ordre de convergence

FISTA, tout comme l'algorithme de Nesterov, atteint une convergence optimale de taux $\frac{1-\sqrt{r}}{1+\sqrt{r}}$ pour un problème fortement convexe, mais avec une convergence asymptotique linéaire selon [TBZ16].

Dans notre cas convexe, ou bien fortement convexe avec une constante de forte convexité extrêmement petite, la convergence est optimale et sera donc d'un ordre de $1/k^2$.

2.8.4 Avantages/désavantages

FISTA est extrêmement simple, peu coûteux, et l'accélération par rapport à ISTA est relativement gratuite. De plus, la méthode ne nécessite pas de recherche linéaire conventionnelle, elle est donc très économique. Cette méthode générale de FISTA ne gère pas les contraintes de bornes, mais nous adapterons au prochain chapitre l'adaptation évoquée

dans [BT09a] au cas de la TEP. Le plus difficile reste de trouver la constante L. Même si nos fonctions objectifs sont convexes, le modèle de vraisemblance logarithmique est théoriquement impossible à résoudre. En effet, la constante Lipschitz pour cette modélisation est infinie. Cependant, localement et loin de zéro, cette constante est finie. On verra qu'en choisissant un bon L a posteriori, la performance maximale que peut offrir FISTA est fortement intéressante en terme de qualité, de vitesse de convergence et de ressources nécessaires. Des méthodes ont été développées afin de trouver un bon L localement, voir [BT09a], plus précisément : le backtracking algorithm.

2.8.5 Bornes supérieures sur L

Pour la modélisation des moindres carrés, on sait que le L optimal vaut deux fois la valeur propre maximale de A^tA. Cependant, A^tA est immense, et non creuse. Si une constante de Lipschitz est demandée dans un cas comme la TDM pour résoudre un moindres carrés, une borne supérieure peut-être donnée par :

$$2\max(A^t(A\mathbb{1})). \tag{2.10}$$

Cette approximation découle directement du théorème des disques de Gerschgorin, car la valeur propre maximale est nécessairement contenue dans l'intervalle créé par l'union des disques de Gerschgorin. Ces disques sont centrés sur chacun des éléments diagonaux de A^tA, et ont pour rayon la somme des autres éléments de leur ligne ou colonne. Ainsi, le maximum atteint parmi tous les disques sera nécessairement plus grand ou égal à la valeur propre maximale de notre matrice, et ce maximum est en fait le maximum des sommes sur les lignes de notre matrice symétrique A^tA.

CHAPITRE 3

Notre adaptation de FISTA pour des problèmes de reconstruction tomographique

En 2009, Beck et Teboulle ont introduit FISTA à l'aide de deux articles : [BT09b] et [BT09a]. Le plus récent de ceux-ci introduisait une application directe de FISTA aux problèmes de débruitage et de défloutage. Tout d'abord, ils ont introduit MFISTA, une version Monotone de FISTA.

Algorithme 7 : MFISTA

Données : Borne supérieure $L \geq L(f)$ (la constante Lipschitz de ∇f)
Résultat : x^*

1 initialisation : $y_1 = x_0 \in \mathbb{E}, t_1 = 1$;
2 *dans notre cas,* $\mathbb{E} = \{x | x_i \geq 0, \forall i\}$;
3 **tant que** $k < K$ **faire**
4 $\quad z_k = prox_L(y_k)$;
5 $\quad t_k + 1 = \frac{1+\sqrt{1+4t_k^2}}{2}$;
6 $\quad x_k = \arg\min F(x) : x = z_k, x_{k-1}$;
7 $\quad y_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1})$;
8 $\quad k = k + 1$;

Comme on peut le voir, cette version nécessite un peu plus de ressources que la version régulière de FISTA, puisqu'elle nécessite d'évaluer la fonction objectif à l'étape six. L'autre contribution très importante pour notre cas est l'adaptation du modèle de moindres carrés régularisé avec variation totale à l'aide d'une combinaison entre MFISTA et d'un algorithme de gradient projeté FGP :

Algorithme 8 : FGP

Données : b–image observée ;
λ –Paramètre de régularisation, N –Nombre d'itérations.
Résultat : x^*

1 initialisation : $(r_1, s_1) = (p_0, q_0) = (0_{(m-1)\times n}, 0_{m\times(n-1)}), t_1 = 1$;
2 **tant que** $k < N$ **faire**
3 $\quad (p_k, q_k) = P_{\mathcal{P}}[(r_k, s_k) + \frac{1}{8\lambda}\mathcal{L}^t(P_C[b - \lambda\mathcal{L}(r_k, s_k)])]$;
4 $\quad t_k + 1 = \frac{1+\sqrt{1+4t_k^2}}{2}$;
5 $\quad (r_{k+1}, s_{k+1}) = (p_k, q_k) + \left(\frac{t_k - 1}{t_{k+1}}\right)(p_k - p_{k-1}, q_k - q_{k-1})$;
6 $\quad x^* = P_{\mathcal{C}}[b - \lambda\mathcal{L}(p_N, q_N)]$;

Ici, P_S est l'opérateur de projection sur l'ensemble S. L'ensemble $\mathcal{P}$ est l'ensemble des paires de matrices (p, q) telles que : $p_{i,j}^2 + q_{i,j}^2 \leq 1$, $|p_{i,n} \leq 1|$ et $|q_{m,j}| \leq 1$. L'opérateur $\mathcal{L}(p_{(m-1)\times n}, q_{m\times(n-1)})$ retourne une matrice L de dimension $m \times n$ telle que chacun de ses éléments $L_{i,j}$ est la somme du gradient en y de l'élément $p_{i,j}$ et du gradient en x de

l'élément $q_{i,j}$. Puisque cet opérateur n'utilise pas de matrice système, nous considérons que son utilisation est peu coûteuse, et donc l'utilisation FGP est peu coûteuse.

Cet algorithme sert à résoudre le problème primal :

$$\min \|x - b\|_F^2 + 2\lambda TV(x), x \in C \tag{3.1}$$

sous sa forme duale. Or, la solution de (3.1) est le résultat de l'opérateur proximal que nous utilisons à l'étape quatre de MFISTA.

Beck et Teboulle utilisent ainsi FGP-MFISTA afin de résoudre le problème :

$$\min \|\mathcal{A}(x) - b\|_F^2 + 2\lambda TV(x), \tag{3.2}$$

tel que $\mathcal{A}$ est un opérateur de floutage et b est une image bruitée. Ce problème entre en effet dans la classe de problème $F(x)$ vu en (2.8). Or, si l'on remplace l'opérateur de floutage $\mathcal{A}$ par la matrice système en TDM ou en TEP, alors on obtiendrait le problème de moindres carrés décrit au chapitre deux ! Encore plus intéressant, on sait que $F(x) = f(x)+g(x)$ nécessite que $f(x)$ soit convexe. Or, nous avons mentionné plus tôt [SV82] que la fonction de vraisemblance logarithmique est elle aussi convexe. On peut donc appliquer FGP-MFISTA directement à nos deux modèles.

3.1 Bornes

L'avantage principal de cette méthode est l'utilisation de FGP lors du calcul de l'opérateur proximal, car il gère les bornes de non-négativité. Ainsi, le code utilisé est initialisé avec 0 comme borne inférieure sur nos variables.

3.2 Autres avantages de la méthode

FGP-MFISTA admet le même ordre de convergence en pire cas que FISTA, qui est optimal. On s'attend donc à de très bons résultats de cette méthode lorsqu'appliquée aux problèmes de TEP et de TDM. De plus, cette méthode ne nécessite qu'une évaluation de fonction objectif supplémentaire par itération, et c'est seulement si l'on veut utiliser la version monotone. Ainsi, à l'aide d'une projection supplémentaire par itération, nous pouvons avoir la monotonicité, mais que nous l'utilisions ou pas, nous avons un algorithme FGP-FISTA ou FGP-MFISTA qui gère les bornes, qui est optimal en pire cas et qui est très économe. L'article [TBZ16] affirme que FISTA appliqué à un problème de LASSO peut être accéléré si remplacé par ISTA à un certain moment, puisqu'ISTA aurait une convergence asymptotique linéaire légèrement supérieure à celle de FISTA. Il serait intéressant d'approfondir cette piste pour notre cas dans un travail futur.

CHAPITRE 4

Méthodologie pour les tests

4.1 Contraintes de Non-négativité

De tous les algorithmes décrits, seuls trois tiennent compte naturellement des contraintes de bornes simples dites de type boîte (box-constraint) : FGP-MFISTA, L-BFGS-B et MLEM. Il nous fallait donc une méthode qui s'ajoutait simplement aux autres algorithmes, afin de pouvoir les comparer ensemble. Notre choix de la méthode s'est arrêté sur la barrière logarithmique. Soit le problème :

$$\min_{x, c(x) \geq 0} f(x), \tag{4.1}$$

alors la barrière logarithmique est définie comme suit :

$$P(x; \mu) := f(x) - \mu \sum_{i \in \mathcal{I}} \log(c_i(x)). \tag{4.2}$$

Il est bien connu que les minimiseurs $x(\mu)$ de (4.2) lorsque μ tend vers zéro approchent

des minimiseurs de (4.1). Par manque de temps, cette technique n'a pas pu être utilisée à son plein potentiel. Nous avons donc choisi de tenter l'heuristique suivante : trouver un μ fixe capable d'assurer la non-négativité des éléments de l'image, afin que le logarithme de Ax reste toujours positif dans notre fonction de vraisemblance logarithmique. Devant le succès de cette méthode, nous avons tenté brièvement d'implanter une technique de réduction constante du paramètre μ à toute les k itérations, et les images obtenues en étaient améliorées. Ces résultats seront dans le test en 5.4.

Ces derniers résultats nous font espérer qu'une technique robuste et bien implémentée de barrière logarithmique pourrait en effet être bénéfique pour les algorithmes ne gérant pas les bornes naturellement.

4.2 Matrice système

4.2.1 En TDM

La matrice système utilisée pour la TDM dans ce document est le résultat d'un stage de recherche fait en 2015 et sera donc mise en annexe. Originellement, le code était en SCILAB, mais fut traduit en Julia. La matrice représente l'acquisition TDM selon un scanner par acquisition à pas discret, avec rayons en cônes. Pour nos problèmes en petites dimensions, notre matrice fais 500 mégaoctets.

4.2.2 En TEP

Comme expliqué plus tôt, la matrice utilisée en TEP est issue d'un programme maison et modélise une version simplifiée de l'aspect géométrique 2D du problème de TEP. Pour nos problèmes en petites dimensions, notre matrice fais 500 mégaoctets.

4.3 Mires utilisées

Les mires sont les images que nous tentons de reconstruire.

4.3.1 Shepp-Logan

Pour nos tests de reconstruction en TDM, nous utilisons la mire de Shepp-Logan avec un crâne visible.

Figure 4.1 – Mire Shepp-Logan pour la TDM

Nous devons cependant enlever ce crâne pour la TEP. La raison est que les os atténuent beaucoup les rayons X, mais que les traceurs radioactifs généraux ne vont pas dans les os. Donc à moins d'utiliser un traceur spécifique, il ne devrait pas y avoir de concentration dans les os en TEP, et donc le crâne n'apparaîtrait pas avec un blanc aussi clair.

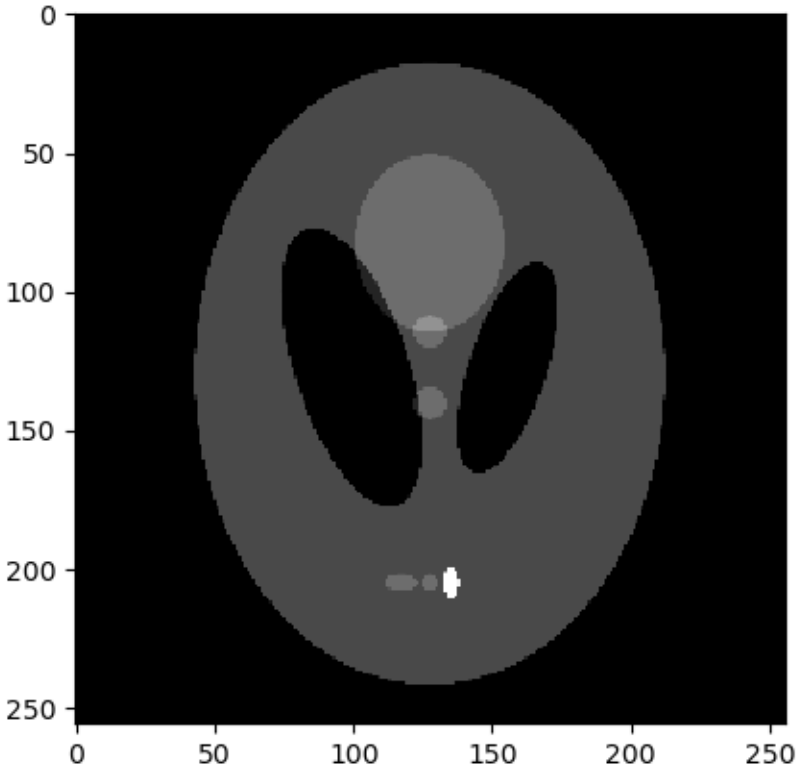


Figure 4.2 – Mire Shepp-Logan pour la TEP

Ces mires étaient intéressantes car elles permettent de déceler des concentrations de différentes intensités et de différentes tailles. La zone d'intérêt de la mire de Shepp-Logan est mise en évidence dans la figure 4.3 :

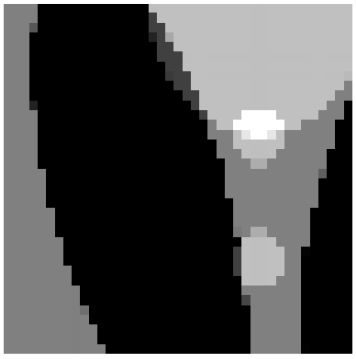


Figure 4.3 – Région d'intérêt de la mire de Shepp-Logan pour la TEP

Cette zone offre beaucoup de détails et de contrastes différents, d'où ce choix.

4.3.2 Deathstar

Cette mire est beaucoup plus simple, elle représente un plan d'une tige traversant une sphère.

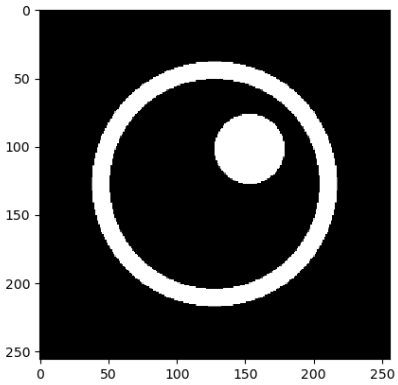


Figure 4.4 – Mire Deathstar

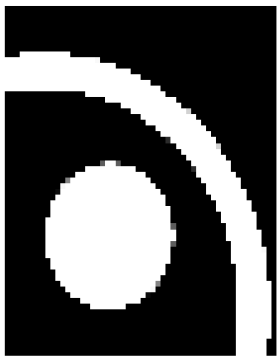


Figure 4.5 – Région d'intérêt de la mire de Deathstar pour la TEP

La zone d'intérêt pour cette mire est mise en évidence dans la figure 4.5 :

Cette zone est très bien désignée pour tester le flou que la variation totale pourrait causer, considérant le rapprochement de deux structures disjointes à très forte intensité.

L'utilisation de deux mires ayant des complexités aussi différentes sert aussi à un autre test. Il nous semblait intéressant de tester la stabilité de l'ordonnancement des algorithmes par rapport à la complexité du problème à résoudre.

4.4 Langage utilisé

Pour ce travail, le langage utilisé est Julia [BKSE12]. Julia est un langage en accès libre datant de 2012. Au moment d'écrire ces lignes, le langage venait tout juste de passer à la version 1.0. Julia représente un très bel équilibre entre les langages plus optimisés comme C, difficiles à utiliser optimalement mais très rapide, et les langages scriptés comme Python, facile à utiliser mais plus lent lorsque utilisé dans leur plus grande simplicité. Julia utilise la technologie de compilation à la volée (Just In Time compiling en anglais),

ce qui lui permet de rejoindre les performances de C lorsque très bien utilisé, tout en ayant une syntaxe aussi simple que Matlab ou Python. Le nombre d'utilisateur de ce langage est en ascension fulgurante, avec plus de 1 800 000 téléchargements en janvier 2018. La liste de packages qui deviennent officiels est de plus en plus grande aussi, et de plus en plus de domaines sont touchés par ceux-ci.

4.4.1 JuMP

Julia a son propre langage de modélisation pour l'optimisation mathématique, qui s'appelle JuMP (Julia for Mathematical Optimization) [DHL17]. Ce langage est très simple à utiliser et possède une différenciation automatique performante. L'utilisation de JuMP était à la base l'une des raisons principales justifiant le choix de Julia.

4.4.2 NLPModels

Le package NLPModels est disponible gratuitement sur Github au :

https ://github.com/JuliaSmoothOptimizers/NLPModels.jl

Développés par Dominique Orban et Abel Siqueira, les NLPModels sont une version extrêmement simple de créer des modèles de programmation non-linéaire, avec ou sans contraintes, lorsqu'on a des problèmes utilisant des fonctions plus intéressantes. La structure des NLPModels est très rapide à apprendre, et la versatilité de cet outil est pour moi sans limites jusqu'à présent.

4.4.3 Structure des modèles dans NLP

La structure utilisée pour la modélisation de nos problèmes est très simple. Nous créons un modèle d'adhérence aux données tel que les moindres carrés ou la vraisemblance logarithmique, puis un modèle de variation totale, et nous les combinons afin de créer les modèles (1.10) et (1.11) par exemple. JuMP étant limité par rapport à l'utilisation d'algèbre linéaire de manière intuitive, nous avons préféré utiliser les NLPModels pour modéliser des fonctions telles que le $\ln.(Ax)$. (En Julia, les opérations faites composantes par composantes utilisent un point après leur nom.)

Vraisemblance logarithmique

Afin de modéliser nos différents problèmes, nous avons donc utilisé la versatilité des NLPModels comme le montre la figure 4.6 :

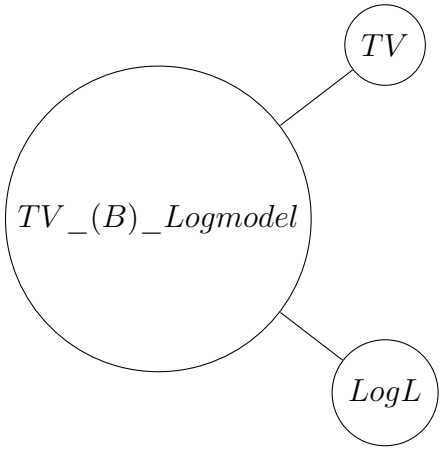


Figure 4.6 – Nous représentons ici le modèle de vraisemblance logarithmique

Le (B) est ici utilisé pour symboliser que certains algorithmes nécessitent d'avoir des outils supplémentaires afin de gérer les bornes de non-négativité.

Moindres carrés

Afin de modéliser nos différents problèmes, nous avons donc utilisé la versatilité des NLPModels comme le montre la figure 4.4.3 :

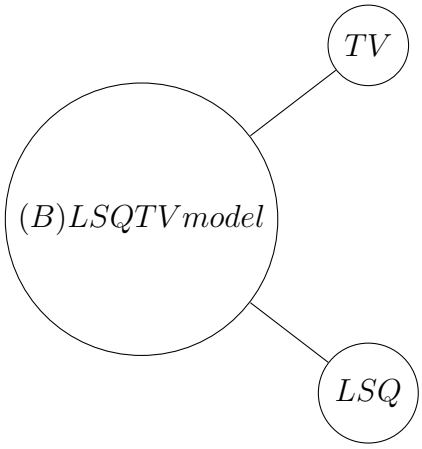


Figure 4.7 – Nous représentons ici le modèle de moindres carrés

4.5 Outil de comparaison

Suite au choix d'utiliser Julia pour nos implémentations, il fallait pouvoir comparer nos algorithmes équitablement. Or, nos algorithmes sont issus de différents packages, faits par différents auteurs dont les capacités en programmation sont variables. L-BFGS-B entre autres est issus d'un code FORTRAN interfacé en Julia. J'ai codé MLEM et L-HZCG et les autres algorithmes sont issus du package LSDescentMethods. Ces différences d'origine des algorithmes suggèrent que de comparer les algorithmes en termes de temps de calcul serait moins crédible. Nous utilisons plutôt une métrique qui se veut plus universelle, soit l'utilisation des opérations plus lourdes. En TEP ainsi qu'en TDM, le plus coûteux est l'utilisation de la matrice système, compte tenu de sa taille. Notre critère de comparaison est donc le nombre de projections ($A * x$ par exemple) et le nombre de rétroprojections

($A' * b$ par exemple). Nos modélisations de la TEP et de la TDM ont en commun que l'évaluation de la fonction objectif coûte une projection, et que l'évaluation du gradient nécessite une projection et une rétroprojection. Nous incrémentons donc de un pour une évaluation de fonction et de deux pour un gradient. Si on évalue une fonction et un gradient en même temps, nous pouvons sauver une projection et on utilisera alors l'équivalent de deux projections.

Les comparaisons de qualité de nos images se fait très bien, voir mieux à l'œil, mais il y aurait tellement d'images résultantes au bout de nos expérimentations qu'il n'est pas réaliste de procéder ainsi. Nous utiliserons plutôt une métrique de comparaison a posteriori. Ayant déjà l'image non bruitée à reconstruire, il est aisé de comparer celle-ci aux images reconstruites et ainsi quantifier la justesse de nos reconstructions. On peut ensuite représenter ces quantifications sur des graphiques pour chaque projection ou rétroprojection utilisée, et on obtiendra un graphique de l'évolution de la justesse de nos images par ressources utilisées. Il est par la suite aisé de compléter notre analyse à l'œil afin de valider les résultats. La métrique utilisée sera le PSNR. Nous avons envisagé d'autres métriques telles que le SSIM et la MSE, mais le PSNR semblait mieux reproduire notre avis par rapport à ce qui est « beau ». Le domaine des métriques de qualité d'images est vaste, et un choix devait être fait, mais notre code contient les autres normes si un lecteur intéressé souhaite pousser l'analyse plus loin.

Les zones choisies afin d'utiliser notre métrique sera appelée la « zone d'intérêt ». Cette zone ciblera une partie de l'image que l'on considère importante, soit des zones contenant des structures ou des contrastes potentiellement problématiques. En effet, il est inutile de tester à quel point le fond de l'image est près de zéro, car en relatif, on veut simplement qu'il soit sombre par rapport au reste de l'image.

Nous avons mentionné que MLEM reconstruit le bruit à partir d'un certain moment dans l'algorithme. Nous verrons un exemple de cette affirmation dans le premier test 5.2. Une

difficulté est donc de trouver quand arrêter l'algorithme. Il serait intéressant de voir si ce problème est commun aux autres algorithmes et si oui, de trouver un bon critère d'arrêt. Comme notre analyse se fait a posteriori, nous verrons si c'est le cas en regardant si le PSNR chute à partir d'un certain moment.

Comme la TDM est structurelle et que la TEP est fonctionnelle, on ne peut pas espérer avoir les mêmes images finales. Il faut donc savoir ce que l'on cherche à obtenir comme qualité pour chacune. La TDM est généralement utilisée afin d'avoir l'image de la structure du corps, et l'on voudra ainsi avoir une image clair et nette. La TEP quant à elle cherche généralement à quantifier des concentrations de traceurs radioactifs judicieusement choisis pour leurs propriétés connues à l'avance. On s'attend alors à ce que les images de TEP soit floues, avec des points plus brillants aux endroits où convergent nos traceurs.

4.6 Obtention des hyperparamètres

Nos tests dépendent de plusieurs hyperparamètres :

1. λ, le paramètre de régularisation
2. L, la constante de Lipschitz de $f(x)$
3. ρ, le paramètre de nos barrières logarithmiques

Pour trouver le meilleur λ pour une fonction objectif, nous bouclons avec un algorithme qui ne nécessite pas d'autres hyperparamètres. En testant plusieurs valeurs de λ avec L-BFGS-B par exemple, il devient facile de garder celui qui nous fait atteindre le meilleur PSNR. Lorsque mesuré, l'effet du lambda ne changeait presque pas d'une mire à l'autre ou d'un algorithme à l'autre. Le lambda variait en fait selon le bruit dans notre image et notre

matrice système. On s'y attendait considérant que λ est un compromis entre l'adéquation aux données et l'a priori anatomique. Comme nos matrices système ne peuvent pas être générées de la même manière, changer le modèle change aussi nos valeurs de λ. Nous avons donc trouvé les λ correspondant à nos modèles pour nos différents niveaux de bruits et ce, pour chacune de nos matrices systèmes. Comme nous avions le même nombre de compte pour 2 mires différentes, il n'est pas très surprenant que l'image à reconstruire n'ait pas eu d'impact notable sur cet hyperparamètre, sans compter que l'image se retrouve dans les deux composantes de notre fonction objectif.

Pour L c'est un peu plus compliqué, puisque cette valeur n'est pas toujours finie. Lorsqu'elle l'est, on la calcule exactement, ou bien on peut l'approximer, par les disques de Gershgorin par exemple. Lorsqu'elle est infinie, si notre fonction est convexe, il est possible de trouver des constantes L telles que nous aurons une approximation locale de notre constante de Lipschitz. Ayant déjà le λ à l'aide de L-BFGS-B, on peut boucler sur différentes valeurs de L et garder celle qui maximise le PSNR.

Pour le ρ, comme nous l'avons expliqué plus tôt dans ce chapitre, nous avons pris une valeur de ρ choisi arbitrairement, qui est réduite selon des pas constants.

4.7 Création du bruit

Afin de rajouter le bruit poissonnien associé au comptage des photons, on utilise une mire, on la vectorise, puis on la pré-multiplie par notre matrice système.

Pour la TEP, on sait que la multiplication d'une ligne de la matrice système par l'image nous donne le nombre de comptes émis sur une projection. En effet, pour chaque élément de l'image, on prend la probabilité que cet élément émette deux photons qui rejoignent les deux détecteurs composant la projection en cour, et on la multiplie par la concentra-

tion dans ledit élément. On obtient ainsi le nombre de comptes produits par cet élément de l'image. Lorsque l'on fait le produit scalaire, on obtient la somme de ce nombre de comptes, pour tous les éléments de notre image. La modélisation du bruit poissonnien passe alors par le tirage d'un échantillon de taille un sur chaque éléments b_j du sinogramme finale b, suivant la loi de Poisson de paramètre $\sum_i A_{j,i}x_i$.

Pour la TDM, c'est différent. En effet, multiplier une ligne de la matrice système par une vectorisation de notre image ne nous donne pas l'énergie détectée. La ligne de la matrice système en TDM contient la distance parcourue dans un voxel, et l'image contient l'information sur la composition dudit voxel. La multiplication des deux, mis en exposant à une exponentielle, nous donne donc l'atténuation du signal lorsque la traversé de l'élément est complétée. Lorsqu'on le fait pour une projection au complet, on n'a qu'à pré-multiplier cette exponentielle par le nombre de photons rayon X utilisé dans la projection pour savoir le nombre de comptes lorsqu'arrivé au détecteur. Par la suite, on sait que cette valeur peut être mise comme paramètre d'une loi de Poisson, et l'on procède comme pour la TEP pour le reste.

Pour nos expérimentations, nous avons décidé d'utiliser plusieurs niveaux de bruit, et donc, plusieurs nombres de comptes différents, puisque le bruit en découle. Pour la TEP, des nombres de comptes totaux de 100 000 et 500 000 ont été utilisés pour des images carrées de 128 voxels par côté. Pour la TDM, nous utiliserons 100 et 1000 photons par projection sous les mêmes conditions. Comme mentionné plus tôt, il est nécessaire de calculer λ pour chaque niveau de bruit que l'on désire tester.

CHAPITRE 5

Résultats

5.1 Objectifs

Le but visé par nos expérimentations est de comparer la performance de différents algorithmes généraux sur des problèmes de reconstruction tomographique. La méthode choisie sera une approche a posteriori, consistant à comparer les images reconstruites avec l'image originale et ainsi de quantifier la justesse de la reconstruction en utilisant une métrique d'image. Nous avons choisi cette méthode afin d'avoir une illustration des compétences de chacun des algorithmes à leur meilleur et ce, pour chaque modèle.

Notre critère d'arrêt sera principalement axé sur le nombre d'itérations. Ce critère fût choisi considérant la difficulté d'obtenir un critère d'arrêt automatique fonctionnel pour tous les algorithmes et ce, pour chaque modèle envisagé. Cependant, certains algorithmes utilisent beaucoup plus de ressources de calcul par itération. Nous mettrons alors un sous-critère basé sur la quantité de ressources utilisées. Comme la vaste majorité du temps de calcul est associée aux projections, nous limiterons nos algorithmes à l'équivalent de n itérations ou bien à $2n$ projections.

Nous irons donc par élimination jusqu'au couronnement d'un ou plusieurs algorithmes en tant que champions.

Pour la TEP, nous comparerons nos algorithmes aux performances de l'algorithme MLEM, puisqu'il est très populaire en pratique. Celui-ci servira ainsi en tant que valeur d'étalonnage. Nous commencerons par montrer comment l'ajout d'un a priori anatomique représenté par la variation totale donne de biens meilleures performances. C'est cette dernière version appelée MLEMTV qui sera utilisée par la suite à titre de comparaison pour nos autres algorithmes.

Toujours pour la TEP, nous ferons un test montrant la performance des différentes formules de gradient conjugué mentionnées au chapitre deux. Comme ces algorithmes ne gèrent par les bornes de non-négativité naturellement, nous utiliserons l'heuristique de barrière logarithmique introduite à la section 4.1. Nous verrons dans ce test l'émergence d'une des quatre formules, et les autres seront ainsi abandonnées pour les tests subséquents. Nous ferons cela par soucis de lisibilité des graphiques et afin de sauver du temps sur les expérimentations. De plus, comme nous n'utilisons pas une méthode robuste de barrière logarithmique, la performance maximale de ces algorithmes ne pourra pas être atteinte et ce test sert donc surtout à montrer qu'il y a un intérêt à tester cette dernière modification pour les algorithmes ne gérant pas les bornes, tel que L-BFGS et la descente du gradient.

Pour la TDM, comme mentionné dans [Ham09], L-BFGS-B peut être considéré comme le meilleur algorithme convergent. Nous l'utiliserons donc pour tester les capacités de l'adaptation de FISTA.

Les autres tests feront varier certains paramètres afin de voir si l'ordonnancement des différents algorithmes est constant ou si certains facteurs peuvent changer cette hiérarchie.

La méthodologie générale des tests est décrite de manière exhaustive dans le chapitre

précédent et ne sera donc pas présentée ici. Par contre, les détails particuliers de chaque test seront disponibles au fur et à mesure par souci de reproductibilité.

5.2 Utilisation d'un a priori anatomique pour l'algorithme MLEM

Le but ici est de démontrer que l'utilisation de la variation totale aide à augmenter la qualité de l'image obtenue par l'algorithme MLEM.

Le test consiste à la reconstruction du Shepp-Logan de dimension 128 pixels par 128 pixels. Le nombre total de comptes dans l'image est d'environ 500 000 et la matrice système est de dimension 73180 par 16384.

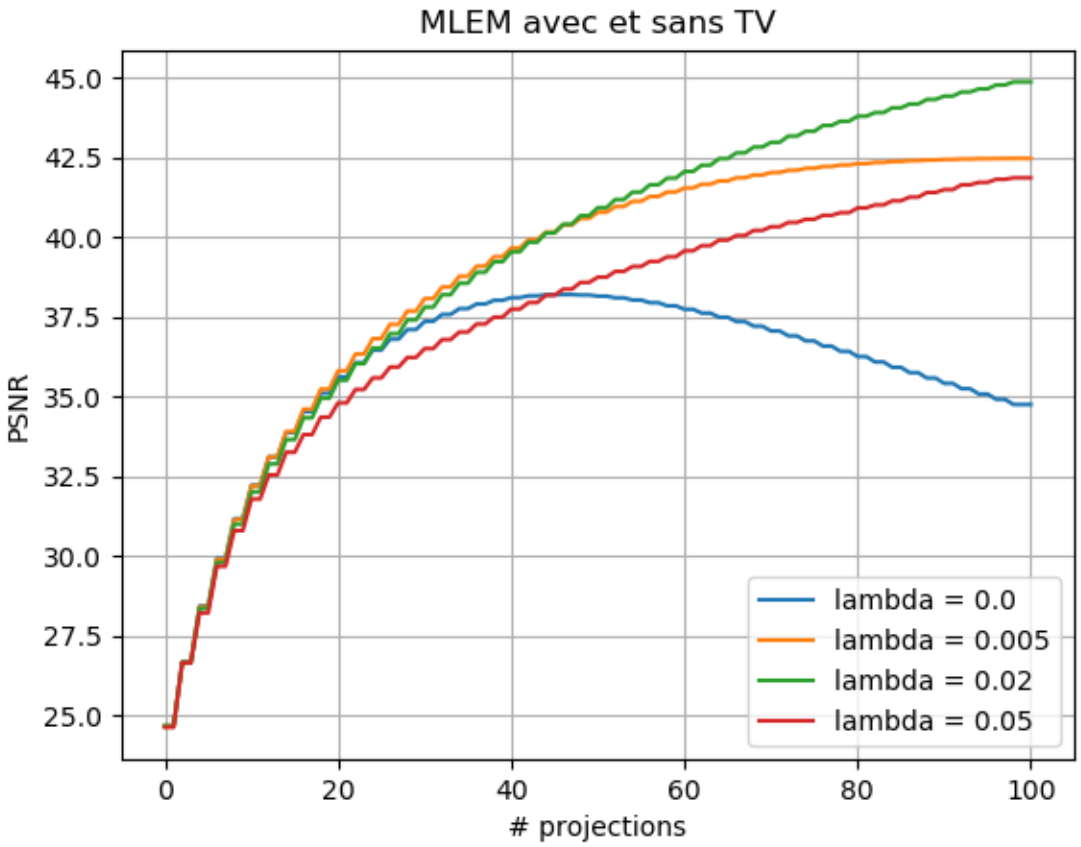


Figure 5.1 – PSNR atteint par MLEMTV en variant λ - Illustration qu'empiriquement, la variation totale appliquée à MLEM donne de meilleures images

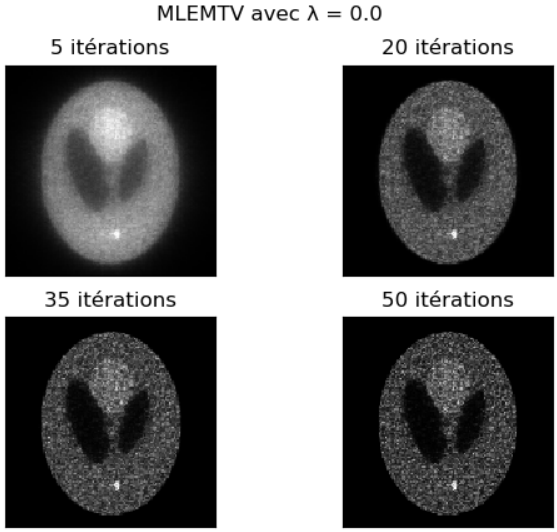


Figure 5.2 – Images reconstruites par MLEM avec $\lambda = 0.0$

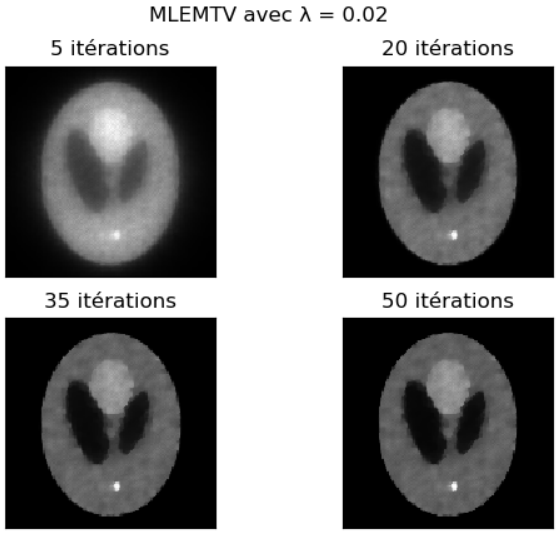


Figure 5.3 – Images reconstruites par MLEMTV avec $\lambda = 0.02$

Sur la figure 5.2, on remarque facilement le très fort niveau de bruit dans la reconstruction de l'image. De plus, le graphe du PSNR de la figure 5.1 indique clairement le point où MLEM commence à reconstruire le bruit. L'ajout de la variation totale ne doit cependant pas se faire aveuglément, car une valeur trop élevée peut polluer les images. De plus, en regardant la formule de MLEMTV, on voit bien qu'une valeur trop élevée de λ nous fera perdre la gestion automatique de la non-négativité des pixels, ce qui fera arrêter l'algorithme brusquement.

Nous concluons donc qu'il faut utiliser MLEMTV, mais qu'une attention particulière doit être portée quant à la valeur de notre paramètre de régularisation λ. De plus, pour une image avec 500 000 comptes, le meilleur λ parmi ceux testé est 0.02, dont on peut voir l'évolution dans la figure 5.3.

5.3 Sélection du représentant de la famille des gradients conjugués

Les tests seront faits sur la reconstruction d'un Shepp-Logan avec 100 000 comptes dans l'image. La matrice système a des dimensions de 73180 par 16384 et le paramètre de régularisation est de 0.015. Nous utiliserons l'heuristique de barrière logarithmique avec un ρ de 3.0, et une réduction par division de 1.5 à toute les 30 projections.

Le problème étant très mal conditionné, nous nous sommes vu obligé de désactiver la mise à jour de Powell [Pow77] lorsque les algorithmes dépassent trois itérations. La condition de Powell n'étant jamais respectée, le gradient conjugué devenait une descente de gradient. Comparer les différentes variantes en devenait ainsi impossible. L'utilisation du gradient conjugué à mémoire limitée de Hager et Zhang, qui utilise un préconditionneur issus d'une matrice quasi-Héssienne dans le sous-espace engendré par les dernières directions

utilisées, semblait être la solution parfaite à ce problème. Cependant, l'implémentation de cette variante est présentement incomplète. Ce dernier algorithme est discuté plus en détail à l'Annexe A.

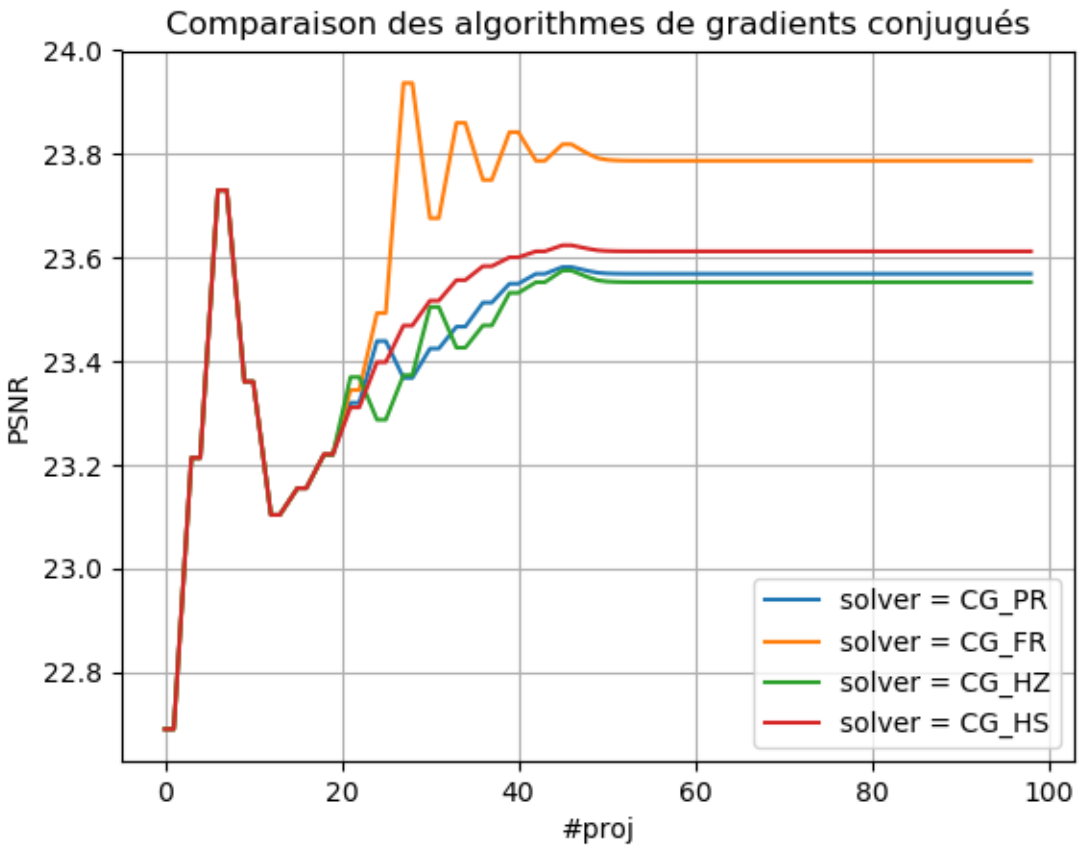


Figure 5.4 – PSNR atteint par nos gradients conjugués - L'algorithme retenu comme gagnant de ce test est la variante de Fletcher et Reeves

Sur la figure 5.4 se trouve le graphe nous montrant l'évolution du PSNR par appel de projections. On peut y voir clairement que la version de Fletcher et Reeves est la plus performante pour notre problème. Les différents tests que nous avons faits corroborent

tous ce résultat. Des images des reconstructions se retrouvent en ANNEXE B. Un lecteur ayant un œil perçant pourra y voir la différence dans le contraste sur les frontières de l'objet dans la mire. En effet, la version de Fletcher et Reeves possède une meilleure distinction entre l'objet et le fond.

Changer la quantité d'information ou la mire n'affecte pas ce résultat, avec l'exception que sur une mire très simple, la version de Hager et Zhang rattrape celle de Fletcher et Reeves et la dépasse à peine. Devant ces résultats empiriques, nous avons décidé d'utiliser CG_FR comme représentant de la famille des gradients conjugués pour le reste de ce chapitre.

5.4 Barrière logarithmique et algorithmes ne gérant pas les bornes

Comme il a été mentionné dans les chapitres précédents, le désir de comparer plusieurs algorithmes issus de l'optimisation générale nous a forcé à trouver une manière de gérer les bornes de non-négativité de notre problème. Le choix s'est ainsi arrêté sur la méthode de la barrière logarithmique. Vu le type de contrainte auquel nous faisons face, l'utilisation de cette technique est très peu coûteuse, ce qui en fait un bon choix.

Cependant, l'implémentation rigoureuse de cette méthode ne fut pas achevée dans les délais disponibles et une simple heuristique est utilisée. Pour ce test, la valeur du paramètre ρ diminuera selon des pas constants. Ce que nous espérions voir dans ce test, c'est que même une gestion aussi simple dudit paramètre apporterait une valeur ajoutée.

Les paramètres choisis pour le test sont une matrice système 18408 par 16384, sur une image 128 par 128 pixels. Le paramètre ρ est initialisé à cinq et il diminue par un facteur 1.5 à toute les 30 projections.

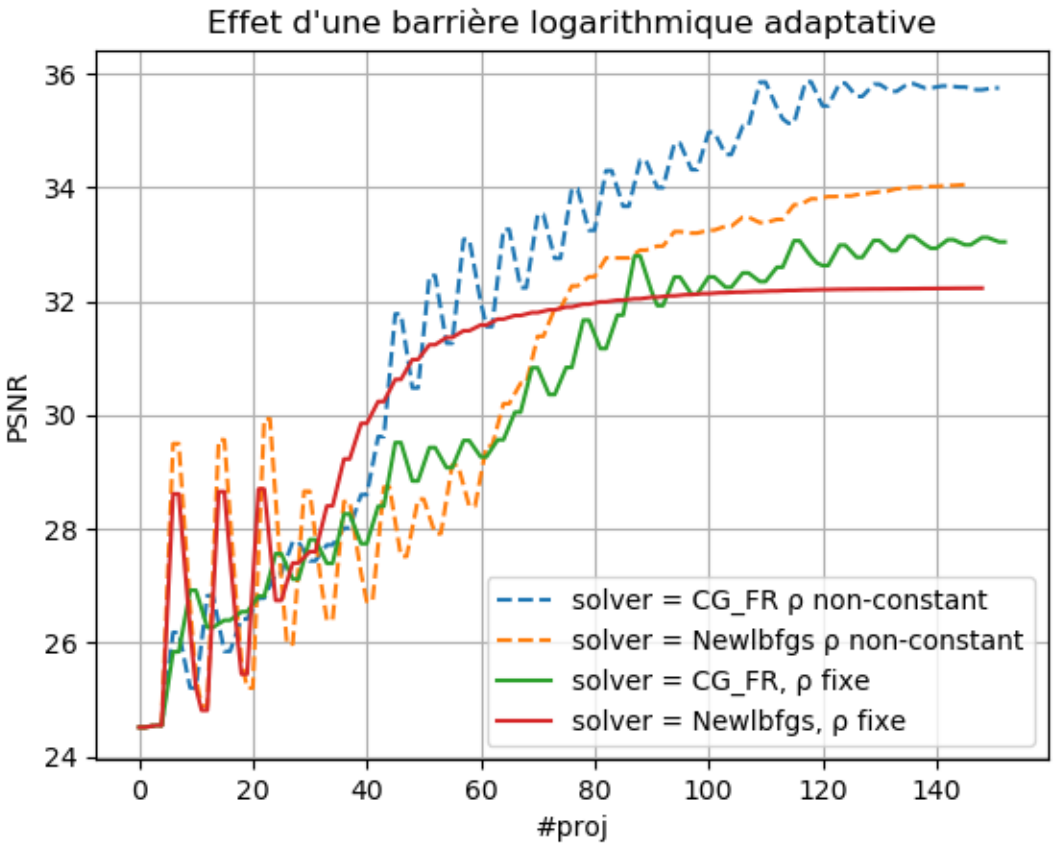


Figure 5.5 – PSNR atteint pour les algorithmes ne gérant par les bornes naturellement - L'utilisation d'une barrière logarithmique variable (en pointillé) augmente la qualité de la reconstruction

Les résultats de la figure 5.4 montrent clairement que la version non-constante atteint un PSNR plus élevé, même si plus lentement pour L-BFGS. Si leur performance en est heureusement améliorée, nous verrons plus tard que ces algorithmes se tiennent encore loin du podium.

Les images reconstruites se retrouvent en Annexe B.

5.5 Effet d'une meilleure initialisation

Dans ce test, nous voulons tenter d'offrir un meilleur point de départ à tout nos algorithmes, de manière économique, afin de les accélérer. Le but était au départ de se rapprocher plus rapidement du minimum de notre fonction, en sachant que des algorithmes allant chercher de l'information du deuxième ordre deviennent très rapide lorsque dans un voisinage de notre solution. Sans surprise, cette piste avait déjà été envisagée et possède plusieurs variantes [TBE+18]. Ces variantes ne diffèrent que sur le choix pris pour générer cette initialisation.

Nous allons donc tester, pour la TEP, l'effet de commencer avec une image de bruit blanc, contre celui de prendre une rétroprojection. Les propriétés de notre matrice système en TEP nous donne en effet que $A^t b$ est un point de départ convenable.

Les paramètres pour ce test étaient un λ de 0.0175, un nombre de comptes de 500 000, une matrice système de dimension 73180 par 16384 et une image de 128 par 128 pixels.

La première chose que l'on remarque dans la figure 5.6, c'est que les algorithmes ne gérant pas les bornes naturellement ne font pas le poids en comparaison à L-BFGS-B, FISTA-FGP et MLEMTV. Cette constatation est générale pour tous les tests que nous avons produit. Nous allons donc abandonner cette famille d'algorithmes pour les tests subséquents.

Comme on le voit dans le graphique, la version de nos algorithmes utilisant une rétroprojection surpasse parfois très légèrement leur version qui ne l'utilise pas, et ce surtout sur les premières itérations. Dans le cas de L-BFGS-B, il est surprenant de voir que ça n'arrive qu'après trente itérations, puisqu'être rapproché du minimum local de la fonction objectif devrait accélérer grandement cet algorithme.

Les images reconstruites détaillées se retrouvent en Annexe B.

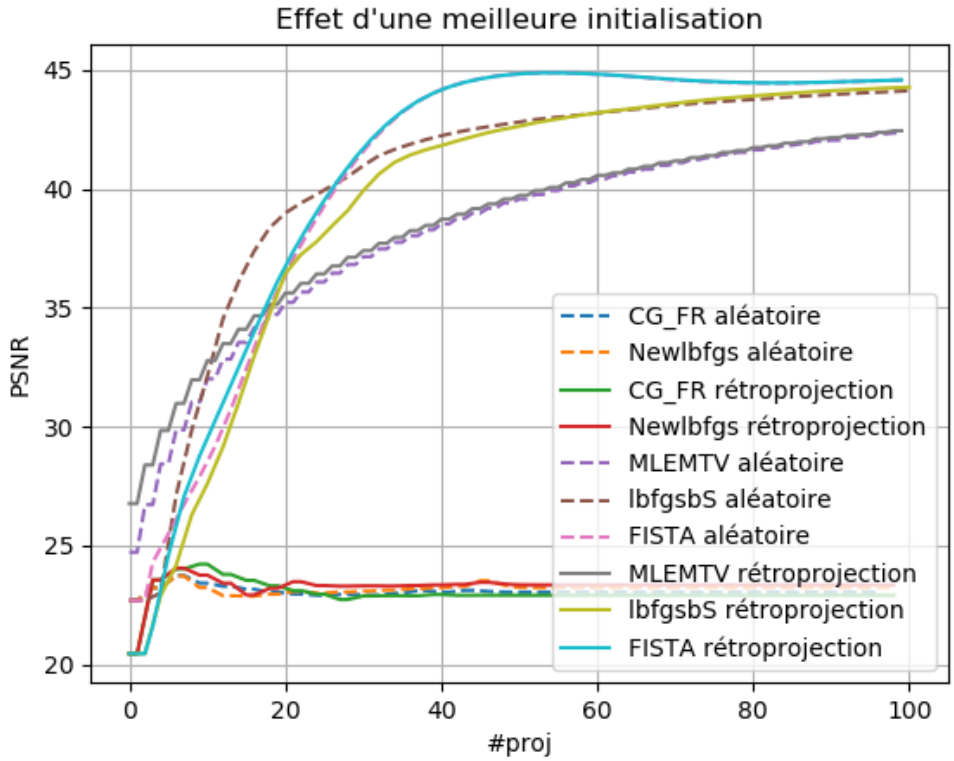


Figure 5.6 – PSNR atteint en variant l'initialisation, avec 100000 comptes - Changer l'initialisation affecte très légèrement nos algorithmes

Le résultat de ce test vient nous confirmer qu'avec une simple projection (l'équivalent de l'ordre du calcul d'une fonction objectif), nous pouvons améliorer légèrement la performance de certains algorithmes. Nous utiliserons donc cette initialisation pour les tests à venir. Il serait intéressant de suivre l'idée de [TBE+18] mentionné plus tôt et d'utiliser un algorithme rapide mais moins précis afin de générer une initialisation différente, pour comparer avec la nôtre.

5.6 Dimentionnalité du problème

La dimentionnalité du problème est un aspect qu'il fallait absolument tester. En effet, à quel point la définition voxelique d'une image affecte la performance de nos solveurs ? On sait par exemple que les algorithmes de type Newton demandent beaucoup de temps de calcul et de ressources mémoires lorsque la dimension augmente, d'où l'utilisation aussi répandue désormais des méthodes de quasi-Newton.

Le prochain test mettra donc nos algorithmes restant à l'essai sur une image de 128 par 128 pixels et une autre de 256 par 256.

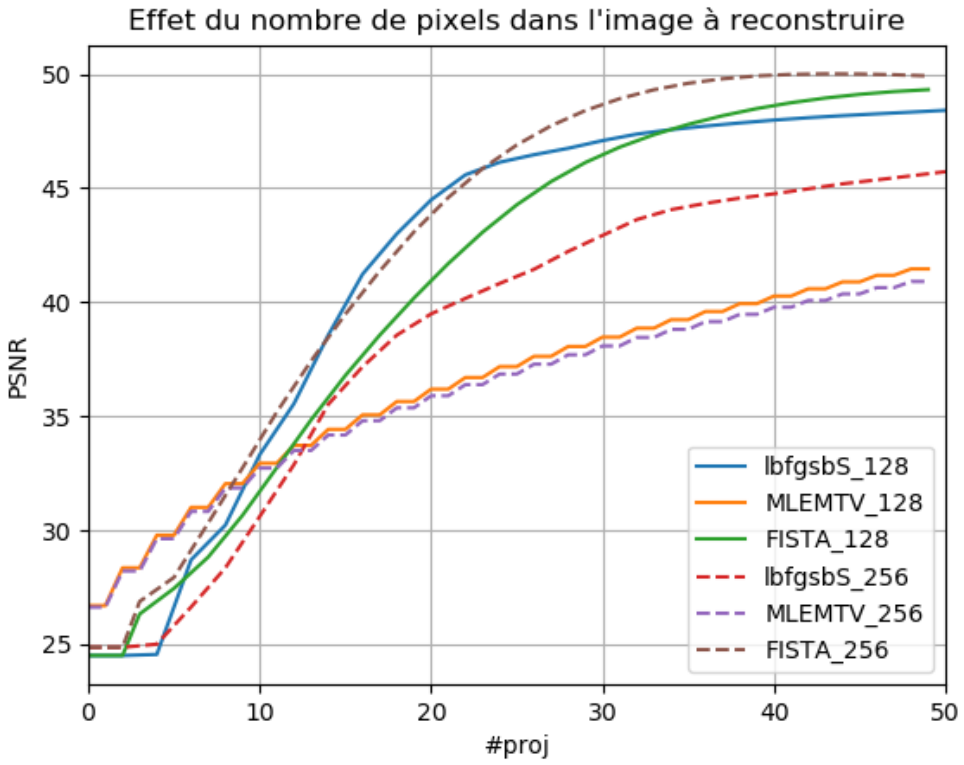


Figure 5.7 – PSNR atteint en variant la définition de l'image, avec 500000 comptes - Varier la dimensionnalité ne fait pas changer l'ordonnancement des solveurs

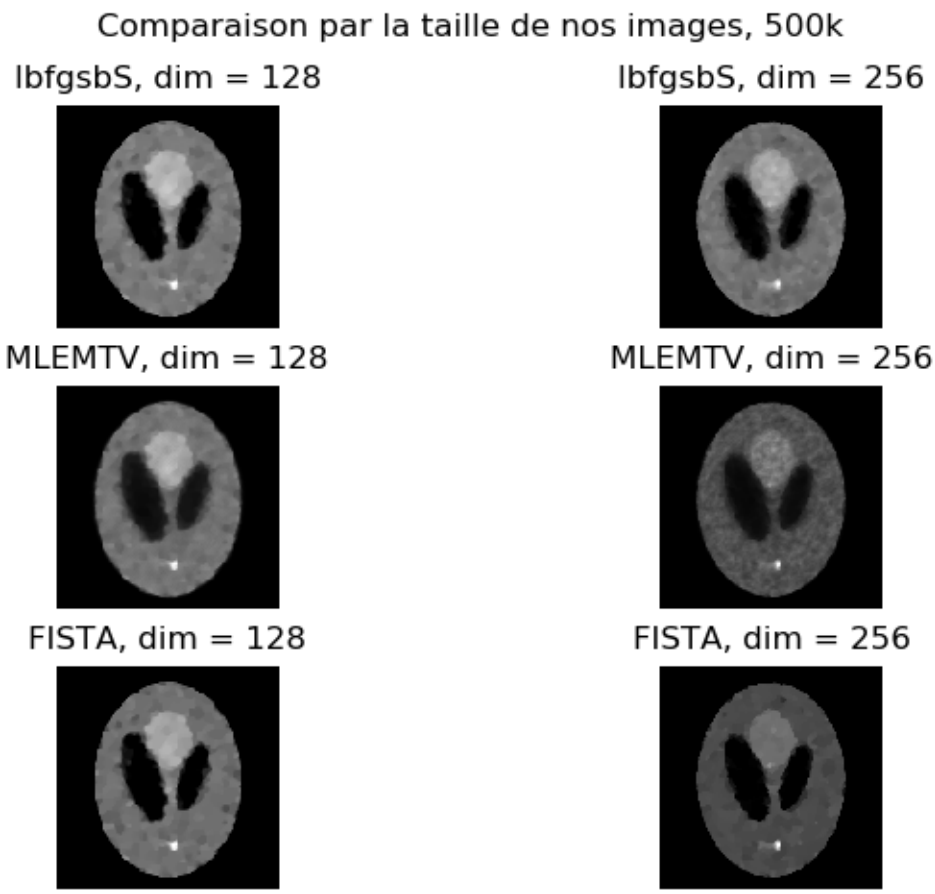


Figure 5.8 – Images reconstruites en variant la définition de l'image, avec 500000 comptes

La figure 5.7 nous montre l'évolution du PSNR sur 50 projections, pour tous les algorithmes restant et ce, pour nos deux matrices systèmes.

Les valeurs de λ ont été choisies empiriquement pour maximiser la performance de chaque algorithme. On remarque que les algorithmes non bornés ayant plus de difficultés, utilisent leur recherche linéaire afin de se sortir du pétrin, ce qui entraîne une augmentation des coûts en projection.

Le graphique montre aussi très clairement la dominance de FISTA-FGP et de L-BFGS-

B sur MLEMTV, non seulement en terme de vitesse mais aussi en terme de précision. Quant au but premier du test, nous remarquons que le nombre de variables ne change pas l'ordonnancement des algorithmes. Or, on remarque bel et bien un petit décalage entre les courbes, mais c'est explicable par le fait que les valeurs de notre paramètre de régularisation ne sont pas parfaites.

5.7 Impact du nombre de projections

Ce test servira à analyser quels algorithmes sont plus ou moins affectés par le nombre de projections que nous utilisons. Comme mentionné plus tôt, on aimerait réduire cette quantité si possible.

Nous prendrons donc nos deux matrices qui représentent des images de 128 pixels par 128 pixels, mais ayant des ratios projections/pixel complètement différents. La première matrice contient 18408 projections, pour un ratio d'environ un pour un, tandis que la plus grande en a quatre fois plus.

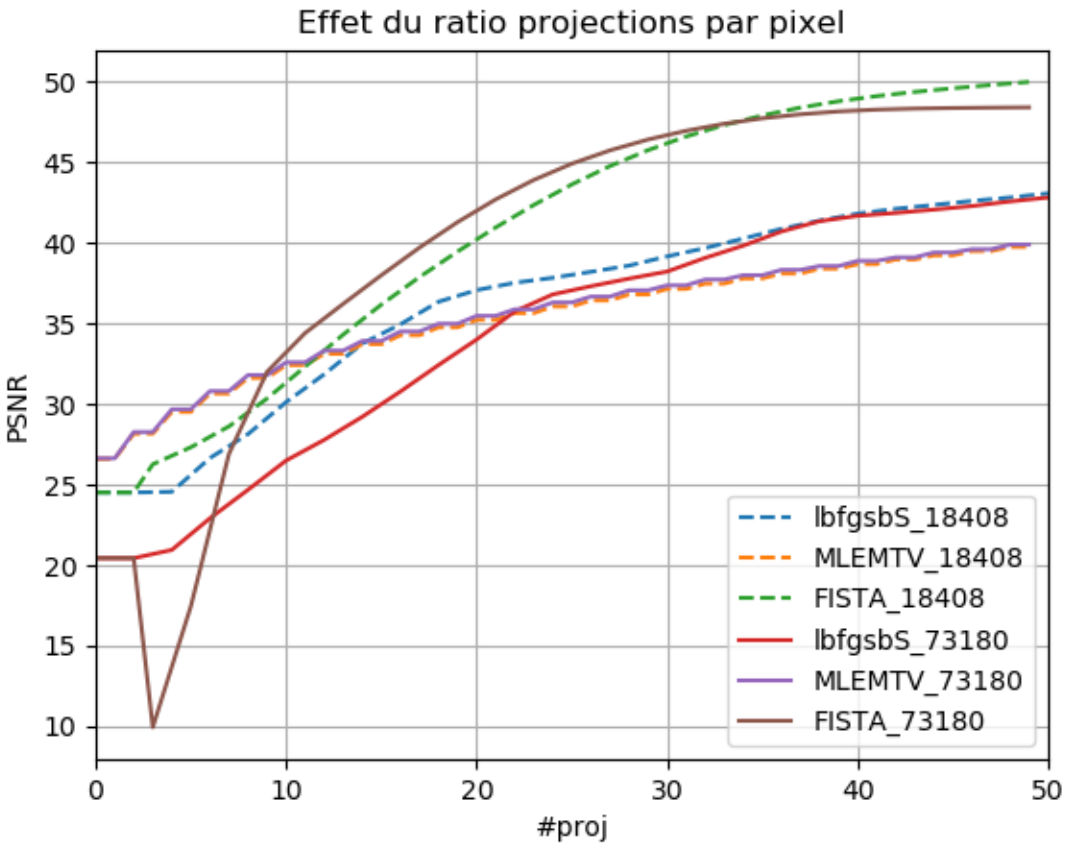


Figure 5.9 – PSNR atteint en variant le ratio projections/pixel, avec 500000 comptes - Le ratio de projections par pixel ne fait pas changer l'ordonnancement des solveurs

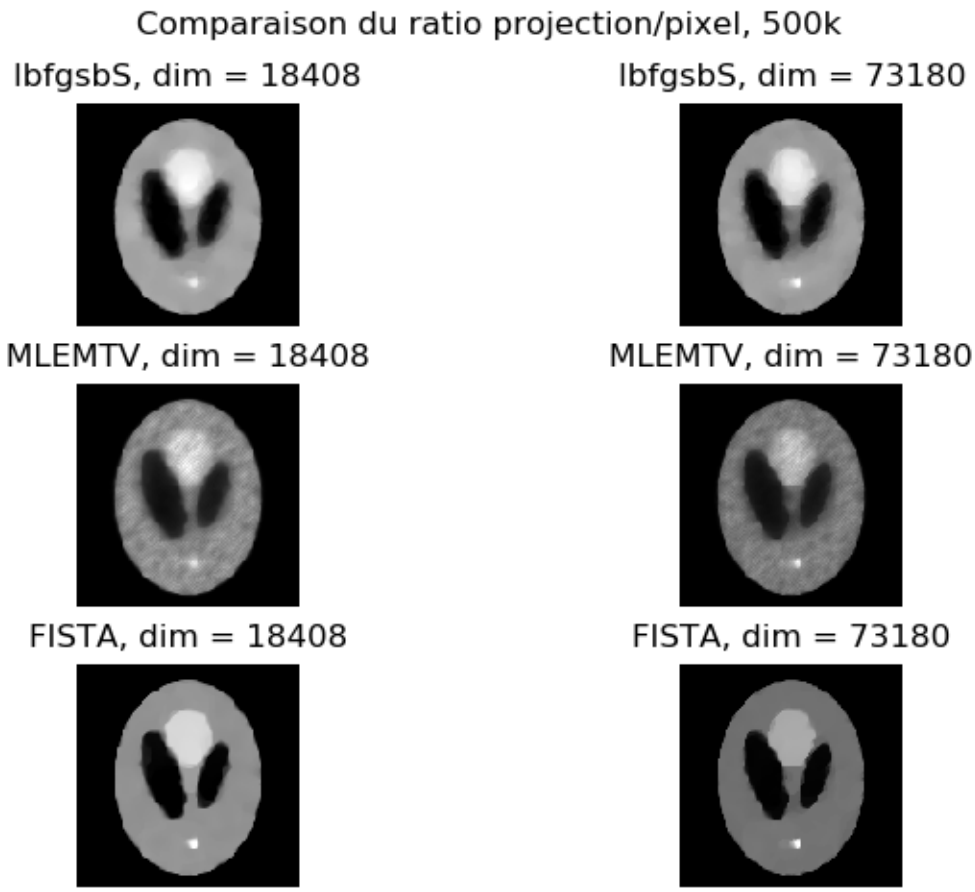


Figure 5.10 – Images reconstruites en variant le ratio projections/pixel, avec 500000 comptes

Ce test apporte des conclusions similaires à celles du test précédent. On peut y voir FISTA-FGP qui trône au sommet, tandis que L-BFGS-B et MLEMTV qui sont deuxième et troisième.

Nous avons tenté de lancer le même test en ayant moins d'information, on compare donc ensuite en utilisant 100000 comptes au lieu de 500000.

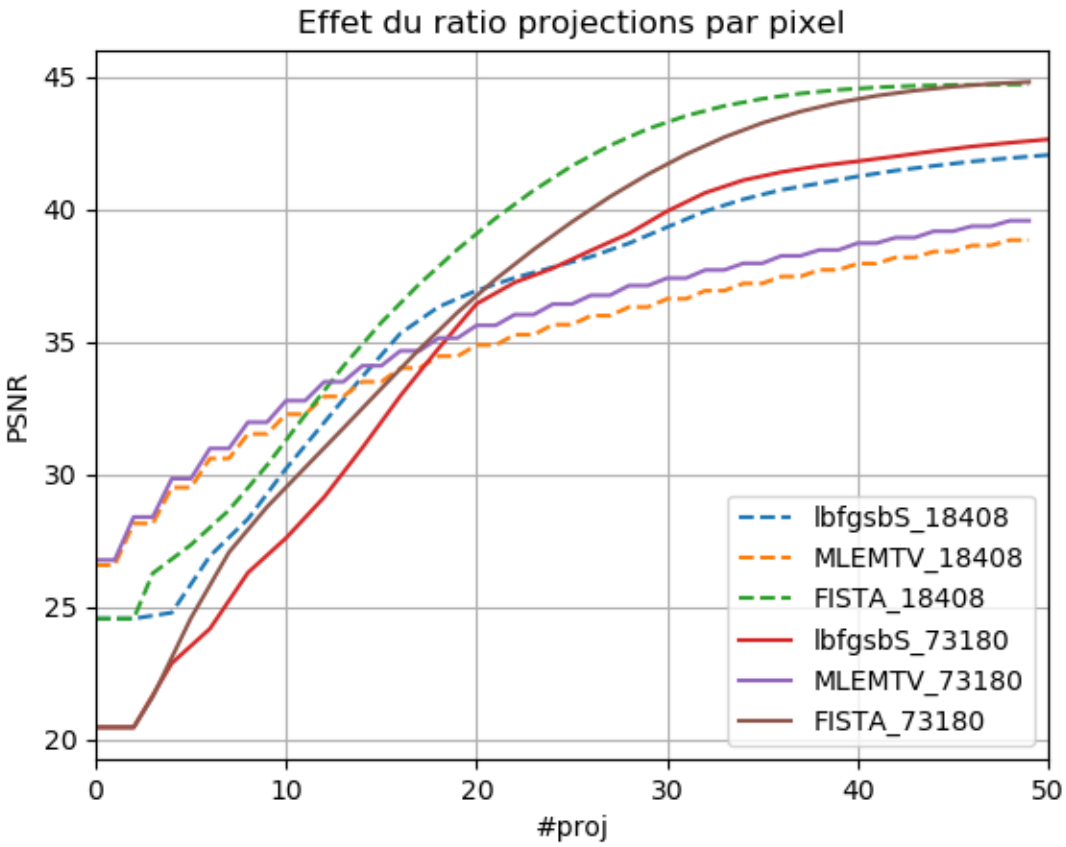


Figure 5.11 – PSNR atteint en variant le ratio projections/pixel, avec 100000 comptes -

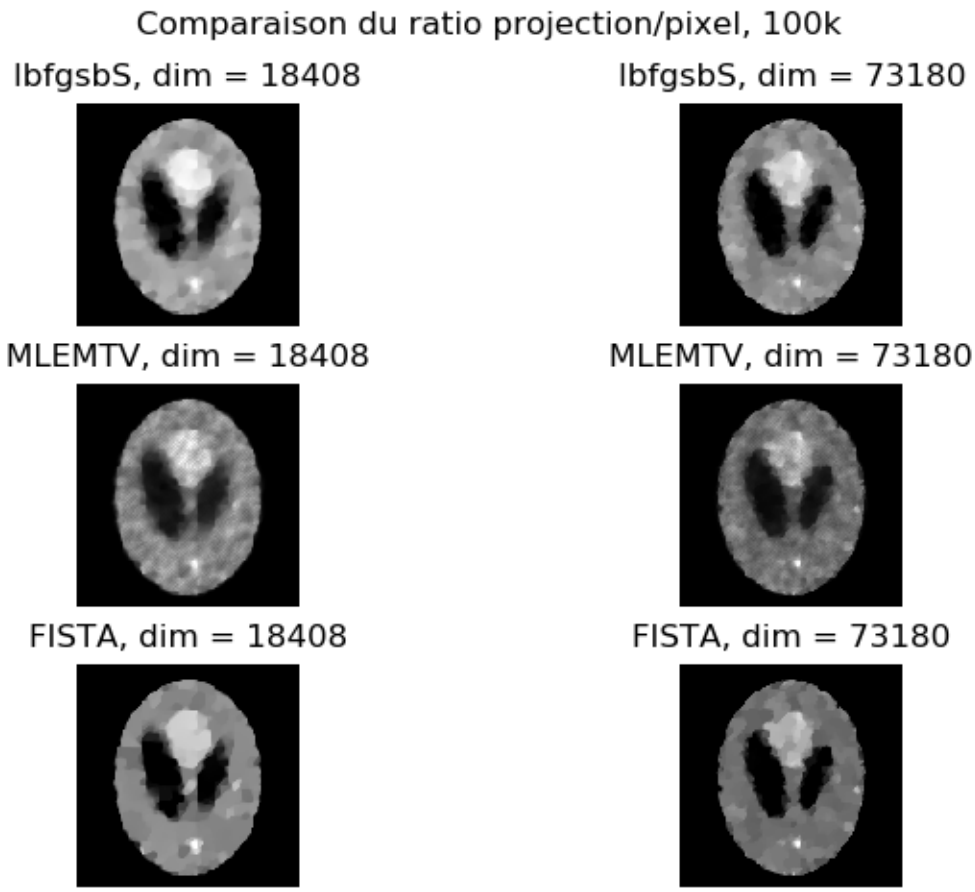


Figure 5.12 – Images reconstruites en variant le ratio projections/pixel, avec 100000 comptes

On remarque encore la même tendance, sauf que L-BFGS-B tend moins vers MLEMTV que vers FISTA-FGP. C'est en fait que MLEMTV est légèrement plus bas qu'avant. On sait par l'article de Hamelin cité plus tôt que L-BFGS-B est très performant en cas de haut niveau d'information. Il semblerait donc normal de voir sa performance chuter légèrement lorsque qu'on baisse ce paramètre. Il est cependant très intéressant de voir que FISTA-FGP et L-BFGS-B gardent une belle performance.

Lorsque l'on regarde la reconstruction des images dont le nombre de comptes est de 100000, on peut remarquer deux artefacts qui sont la conséquence du fort bruit Poissonien. Le premier artefact est une exubérance sur la gauche de l'ellipse la plus haute. Le deuxième se retrouve à la droite de l'ellipse représentant le ventricule de droite. Il est très intéressant de voir que même si nos algorithmes généraux ne sont pas capables de les ignorer, l'algorithme clinique MLEMTV ne le peut non plus.

5.8 Complexité de l'image à reconstruire

Un autre aspect à évaluer est le niveau de détail à reconstruire dans notre image. En effet, on aimerait que nos algorithmes restent performants s'ils sont un jour appliqués à des données réelles. Ne disposant pas d'un tel jeu de données, nous allons comparer les performances sur notre mire Shepp-Logan et sur notre mire DeathStar. La première est un peu plus complexe que la deuxième, qui elle est vraiment très simple.

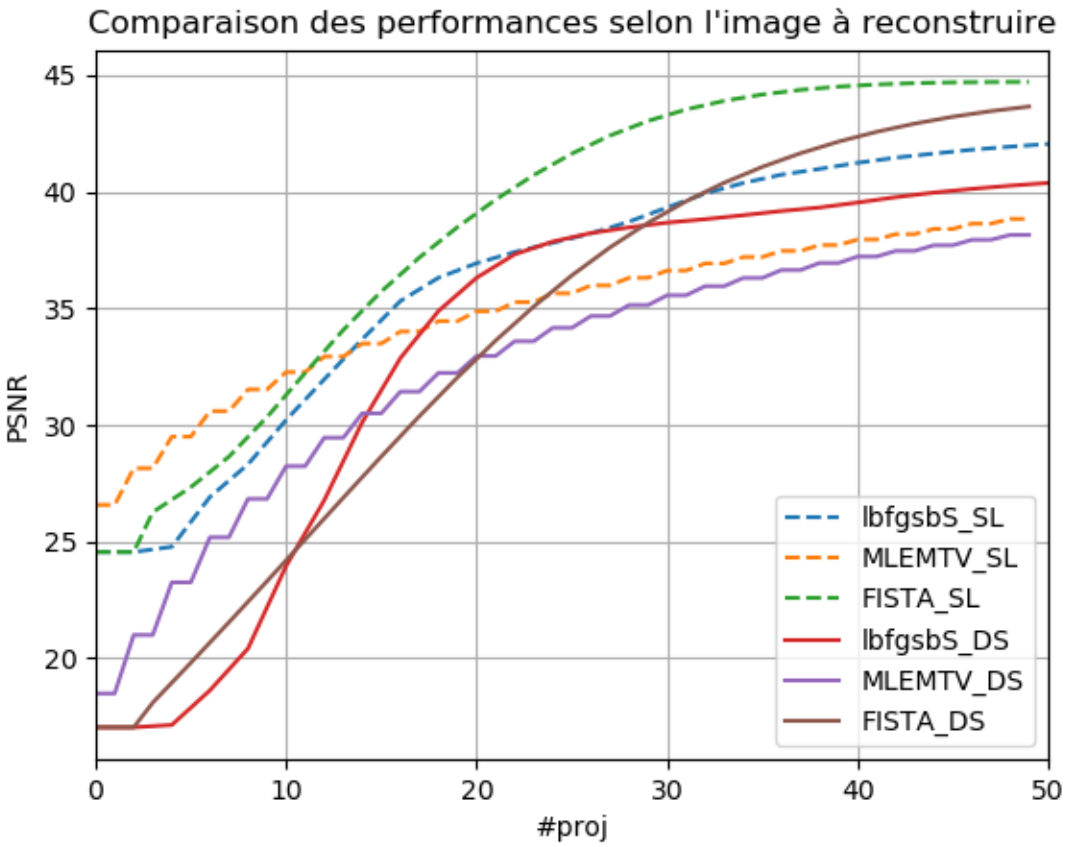


Figure 5.13 – PSNR atteint en variant la mire, avec 100000 comptes - L-BFGS-B semble gagner des points sur des images plus simples (DeathStar)

Encore une fois, comme on le voit sur la figure 5.13, la performance de FISTA-FGP est très impressionnante. On remarque que L-BFGS-B semble gagner un peu de vitesse lorsque sujet à la reconstruction d'une image plus simple. MLEMTV se défend assez bien avec une complexité plus faible.

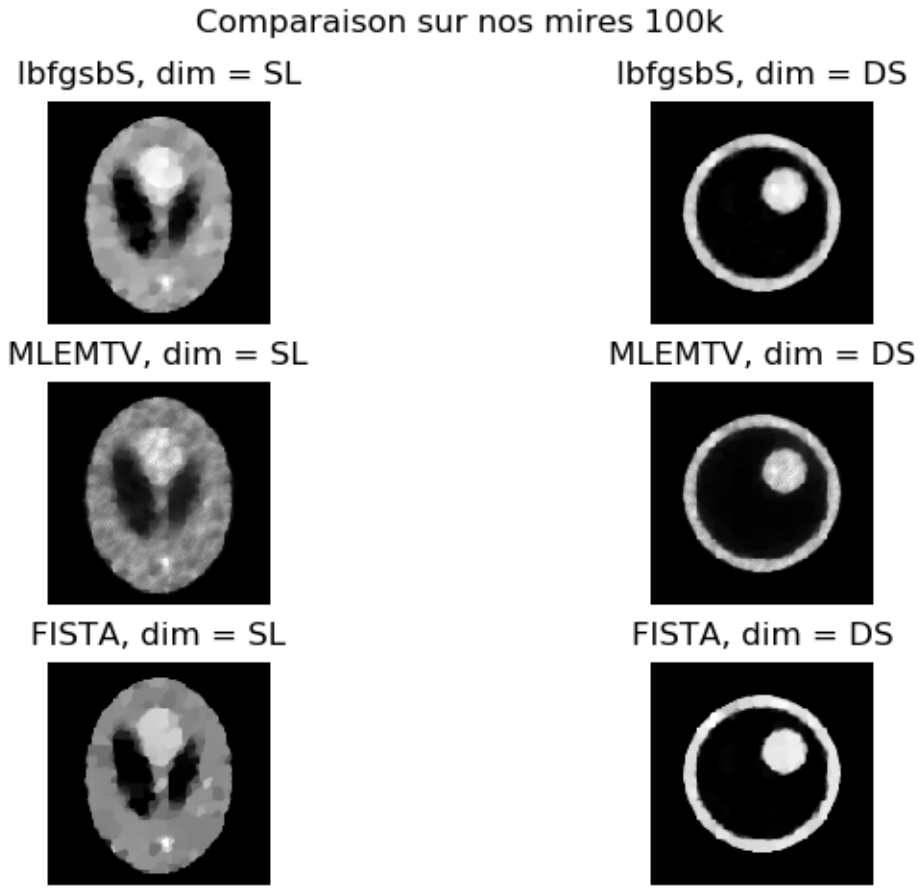


Figure 5.14 – Images reconstruites en variant la mire, avec 100000 comptes

Lorsque vient le temps de comparer les images reconstruites, FISTA-FGP est sans aucun doute le grand gagnant pour ce test.

5.9 Reconstruction LS en TEP

Ce dernier test pour la TEP comparera nos deux modèles, soit la Vraisemblance Logarithmique et les moindres carrés. Pour ce test, nous utilisons une matrice système de

dimension 73180 par 16384. La mire à reconstruire est celle de Shepp-Logan en dimension 128 pixels par 128 pixels. Les λ utilisé pour LL(Log-Likelihood) et LS(Least Square) sont respectivement de 0.0375 et 0.08. Nous avons utilisé 35 itérations et le nombre de comptes était de 100000.

Nous avons choisis 100000 comptes car nous savons qu'avec beaucoup d'information, disons 500000 comptes, le bruit diminue de plus en plus vers un bruit Gaussien et qu'alors, le modèle des moindres carrés devient bien défini. Nous avons donc pris un nombre de comptes beaucoup plus faible afin de voir si les PSNR atteint par nos algorithmes sur le modèle des moindres carrés surpassaient ceux atteint en utilisant la vraisemblance logarithmique.

En examinant 5.15, il est très stimulant de voir la performance de L-BFGS-B pour le LS. En effet, il atteint très vite un bon PSNR, pour ensuite stagner à un niveau légèrement inférieur à FISTA-FGP sur les deux modèles. On remarque encore une fois que MLEMTV reste plus bas que les autres.

Ces résultats sont intéressants car en théorie, le LS ne représente pas parfaitement le phénomène physique que nous mesurons. Cependant, en pratique, ces images semblent aussi précises que celles obtenues par le modèle de LL.

5.10 Utilisation de MFISTA

Comme nous l'avions mentionné, il y a une version monotone de FISTA, dénommée MFISTA, qui est disponible. Dans une application normale, MFISTA nécessite de calculer une fonction objectif de plus par itération. Comme FISTA calcule normalement le gradient de l'équation aux données, on ne peut pas utiliser une fonction calculant l'objectif et le gradient d'un coup. Ainsi, la fonction issue des NLPModel, «objgrad !», n'est pas

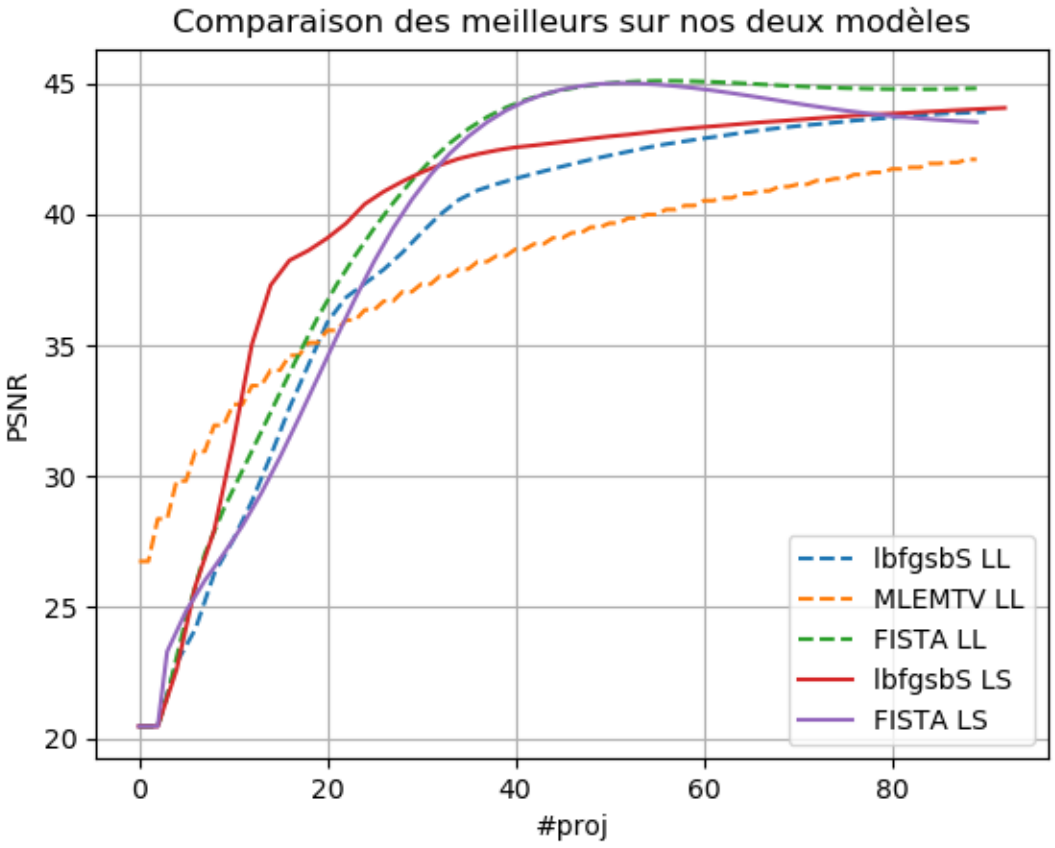


Figure 5.15 – PSNR atteint par nos meilleurs algorithmes, sur nos deux modèles - La modélisation des moindres carrés donnent des reconstructions valides

utilisable pour sauver le coût de la projection supplémentaire qu'implique l'utilisation de cette méthode monotone.

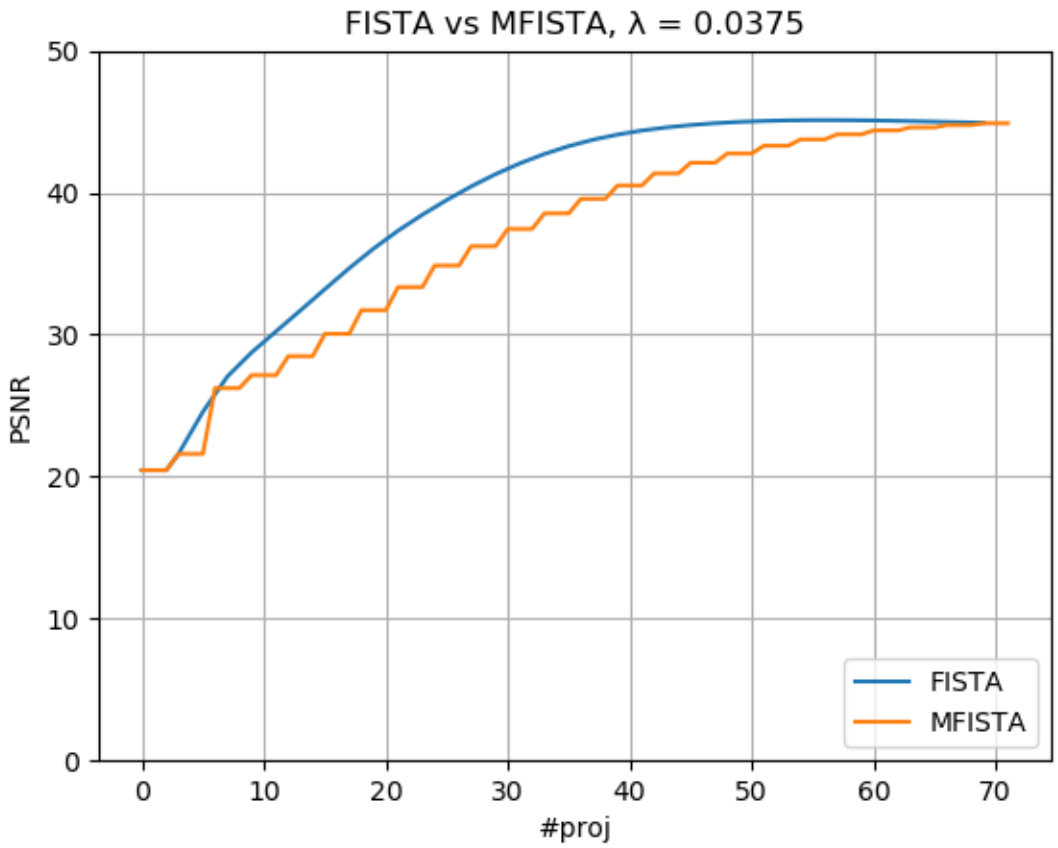


Figure 5.16 – PSNR atteint par MFISTA et FISTA - On voit que MFISTA atteint le même PSNR que FISTA, mais plus lentement

Comme on peut le voir dans la figure 5.16, le même PSNR est atteint, mais beaucoup moins vite. Cette conclusion est la raison nous ayant fait choisir FISTA comme méthode à tester pour cet ouvrage.

5.11 Reconstruction LS en TDM

Comme mentionné au début de ce chapitre, nous voulions appliquer FISTA-FGP à un problème de moindres carrés régularisé, puisque cela entre dans la gamme de problème pour laquelle FISTA-FGP a été développé. Comme L-BFGS-B est l'étalon pour la TDM avec beaucoup de comptes, une comparaison avec celui-ci sera faite.

Le test sera fait avec une matrice système générée de 161200 par 16384, pour une image de 128 pixels par 128 pixels. 80 itérations sont utilisées, mais c'est pour illustrer que le PSNR se stabilise après seulement quelques itérations. Cette propriété indique que les algorithmes sont rapides et qu'un critère d'arrêt sophistiqué pourrait être utilisé, voir [HGD].

Le test s'avère très révélateur. En effet, en plus du fait que le problème en CT est mieux défini en général, il nous permet aussi de mettre une borne supérieur sur la valeur de nos pixels. Nos variables, lorsqu'on utilise FISTA-FGP et L-BFGS-B, peuvent ainsi profiter d'un domaine entre zéro et un. On voit dans la figure 5.17 que les deux algorithmes sont rapides et atteignent des qualités d'images plus que respectables, ce qui est confirmé dans la figure 5.18.

Bien que d'autres tests étaient prévus pour le CT, le manque de temps a joué contre nous. Cependant, nous tenions vraiment à démontrer que FISTA-FGP était valable et qu'il mérite de se voir accorder une plus grande utilisation en pratique.

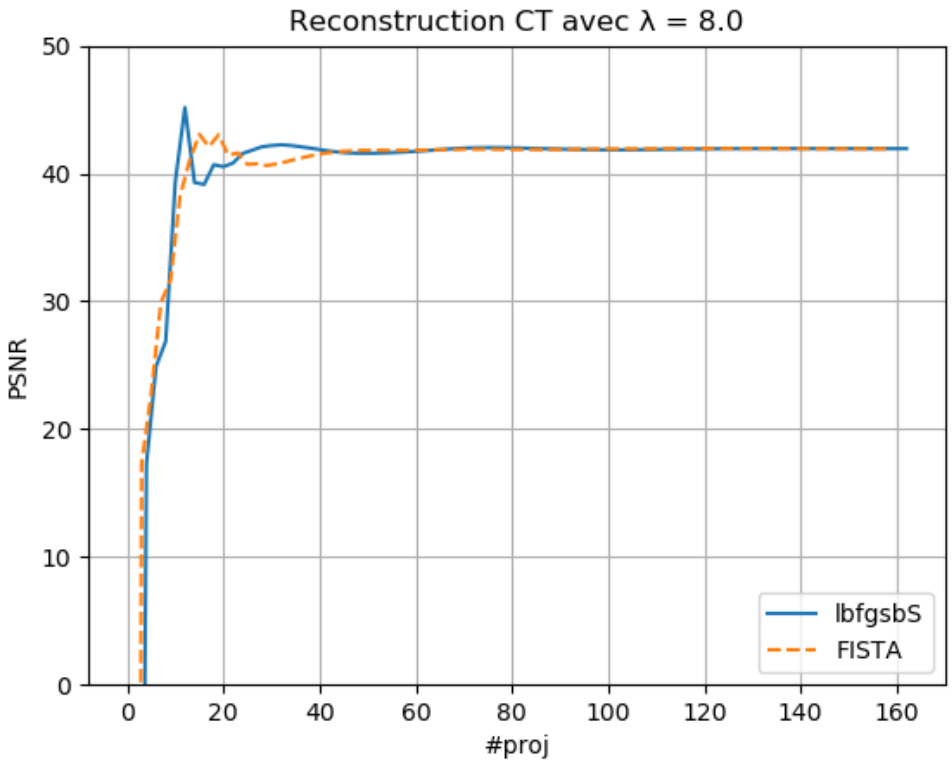


Figure 5.17 – PSNR atteint avec L-BFGS-B et FISTA-FGP en TDM

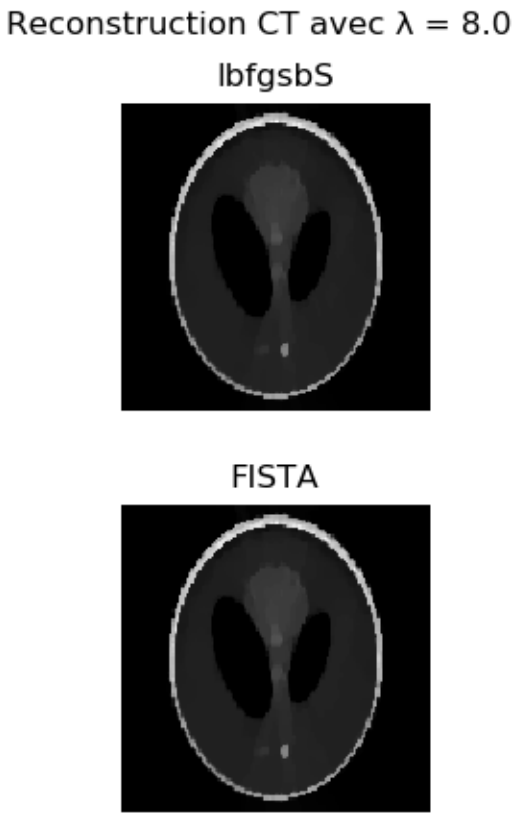


Figure 5.18 – Images reconstruites avec L-BFGS-B et FISTA-FGP en TDM

CONCLUSION

En conclusion, nous avons montré que l'algorithme FISTA peut être performant dans les problèmes de reconstruction tomographique, que ce soit en tomodensitométrie ou en tomographie par émission de positons.

La comparaison avec un algorithme bien reconnu dans ce domaine, MLEM, montre que des recherches plus approfondies devraient être faites pour l'application de FISTA et de L-BFGS-B sur des données réelles pour la TEP. Même si FISTA battait pour ainsi dire toujours L-BFGS-B lors de nos tests, l'impossibilité d'obtenir une constante de Lipschitz de manière analytique pour le modèle de vraisemblance logarithmique nécessiterait de tester la méthode de backtracking mentionnée au chapitre deux. Nous savons cependant que dans le cas des moindres carrés, cette constante existe toujours et est fixe selon la matrice système utilisée.

L-BFGS-B quant à lui n'a qu'un seul paramètre qui lui soit propre, mais varier ce paramètre n'apportait pas de changement dans nos tests empiriques, et la valeur de base fut donc sélectionnée. Nous avons ainsi utilisé un m de cinq. Toujours près des performances de FISTA, L-BFGS-B est un très bon deuxième qui perd une finale chaudement disputé.

MLEM, que l'on sait très populaire en pratique, ne semble pas être capable de suivre ces algorithmes issus de l'optimisation moderne.

Nous avons également brièvement testé une solution pour adapter des méthodes issues de l'optimisation générale, mais ne pouvant traiter les bornes. Les résultats obtenus avec l'heuristique basée sur une barrière logarithmique sont encourageant et justifient de tenter la programmation d'une solution plus rigoureuse.

Un algorithme nouveau, développé par Hager et Zhang, et réputé pour être très performant dans les problèmes mal conditionnés fut étudié mais son implémentation est toujours incomplète.

Perspectives

Les travaux futurs issus de ces tests incluent ainsi le test d'un algorithme robuste de barrière logarithmique pour la TEP, l'implémentation d'une version adaptative de FISTA qui permet de calculer la constante de Lipschitz localement, l'essai de différentes techniques d'initialisation de l'image, l'essai des algorithmes TRON et du gradient conjugué à mémoire limitée de Hager et Zhang, l'adaptation des tests à la reconstruction 3D, l'essai de différentes formes de régularisation et l'exploration des préconditionneurs. De plus, refaire tous ces tests avec des images réelles serait un très bel ajout.

Nous pensons qu'il serait aussi très intéressant d'investiguer les ajouts qui ont été fait à MLEM au cours des années, notamment les Ordered Subsets (OS), et de voir comment les appliquer à FGP-FISTA ou L-BFGS-B.

ANNEXE A

7.1 Adaptation de L-HZCG

7.1.1 Principe de la Méthode

En 2013, Hager et Zhang ont publié un article [HZ13] mettant en vedette une nouvelle variante de leur gradient conjugué, dans laquelle ils utilisent une technique économique et efficace de préconditionnement lorsque nécessaire. Cette variante serait selon leur dire plus rapide que L-BFGS en temps de calcul et même plus économe en terme d'appel de fonctions. Devant ces affirmations, nous trouvions très intéressant de se pencher sur un tel algorithme, considérant qu'il pallierait parfaitement aux défauts de la famille des gradients conjugués présentée au deuxième chapitre de ce mémoire. Cependant, à ce jour, l'implémentation n'est pas aussi efficace qu'escomptée. Il est bien connu que les algorithmes de gradients conjugués sont très sensibles, et il se pourrait bien que notre implémentation en soit affectée. Cependant, beaucoup d'heures de travail ont été investies à la compréhension et l'implémentation de cet algorithme, d'où la raison de sa présence en annexe. Nous pensons que cet algorithme pourrait être très prometteur une fois fonctionnel, et combiné avec une bonne barrière logarithmique, comme mentionnée à la section 4.1.

Hager et Zhang expliquent qu'avec une mémoire limitée de quelques directions précédentes, une vérification de l'orthogonalité de la direction la plus récente avec les directions de la mémoire permettrait d'évaluer la qualité de notre orthogonalité. En cas de perte de l'orthogonalité, un problème dans un sous-espace sera réglé avec L-BFGS, et l'approximation de l'inverse de la hessienne issus de la résolution de ce problème pourra être adaptée et sera finalement utilisée comme préconditionneur.

7.1.2 "Pas assez orthogonale"

La raison pour laquelle l'algorithme du gradient conjugué est aussi sensible au conditionnement, c'est qu'il repose sur la conjugaison des résidus d'une itération à l'autre. Comme la conjugaison est un principe analogue à l'orthogonalité, un mauvais conditionnement vient la contaminer de manière désastreuse. On finit par perdre l'orthogonalité d'une itération à l'autre, et c'est à ce moment que l'on produit un cycle de directions n'étant pas « assez » orthogonale. Ainsi, l'espace échantillonné par nos directions d'une itération à l'autre ne change presque pas, et l'algorithme en est ralentit. Powell [Pow77] avait déjà proposé des techniques de réinitialisation de l'algorithme du gradient conjugué, mais la technique mentionnée par Hager et Zhang est toute autre, car la nouvelle direction issue du l'étape de réinitialisation tient en compte le passé, et préconditionne notre problème.

7.1.3 Explication du sous-espace

Soit le problème de minimisation sous contraintes linéaires :

$$\min_{s.a. Ax=b} f(x), \tag{7.1}$$

on sait que les directions réalisables sont des combinaisons linéaires des colonnes du noyau

de A. Soit Z la base du noyau de A, les conditions d'optimalité d'ordre un de ce problème nous donne que :

$$x^* min\ \ local\ \ de\ \ f \Rightarrow \nabla f(x^*)Z = 0 \tag{7.2}$$

Soit S_k la liste des directions précédentes que nous avons gardées en mémoire, nous allons résoudre le problème suivant lorsque nous considérons que nous perdons l'orthogonalité :

$$\min_{z \in S_k} f(x_k + z). \tag{7.3}$$

Or, les directions réalisables de ce problème reviennent exactement au rôle que jouait Z pour le problème (7.1). On aura en particulier que résoudre (7.3) nous fournira un gradient $\nabla f(x^*)$ orthogonale à toute les directions de S_k, et donc une direction réinitialisée judicieusement choisie. De plus, la résolution de ce problème est très simple considérant la linéarité des directions réalisables. Il est beaucoup plus facile de résoudre cette reformulation du problème :

$$\min f(x_k + \sum_{i=1}^{m} \gamma_i d_{k-i}), \tag{7.4}$$

car elle n'a plus de contraintes à résoudre. Il est donc possible d'y appliquer un algorithme de L-BFGS et de résoudre rapidement ce dernier. L'extraction de l'approximation de la hessienne H_{k+1} se fait alors sans coûts supplémentaires, et est un bon préconditionneur du sous-espace. Une utilisation de l'approximation de Barzilai-Borwein [BB88], que l'on dénote par σ_k peut alors être faite afin d'évaluer un préconditionneur économique pour la partie de l'espace complémentaire au sous-espace en cours.

7.1.4 Préconditionneur

Soit Z une base orthonormée de S_k, Hager et Zhang propose le préconditionneur suivant :

$$P_k = Z\hat{P}_k Z^t + \sigma_k \overline{ZZ}^t, \tag{7.5}$$

la première partie étant une approximation du hessien restreinte au sous-espace, et la deuxième une approximation BB de l'espace complémentaire au sous-espace. Une fois le préconditionneur obtenu, les auteurs affirment qu'une étape de préconditionnement de la direction accélère grandement la convergence. L'application d'un préconditionneur à l'algorithme régulier de gradient conjugué de Hager et Zhang nous donne :

Algorithme 9 : HZ-CG préconditionné

1 **pour** $k = 0, 1, ...$ **faire**
2 $\quad x_{k+1} = x_k + \alpha_k d_k, \quad \alpha_k$ issus d'une recherche linéaire ;
3 $\quad \beta_k = \frac{y_k^t P_k g_{k+1}}{d_k^t y_k} - \theta_k \frac{y_k^t P_k y_k}{d_k^t y_k} \frac{d_k^t g_{k+1}}{d_k^t y_k}$;
4 $\quad d_{k+1} = -P_k g_{k+1} + \beta_k d_k$;
5 $\quad k = k + 1$;

En utilisant la définition géométrique de la projection ainsi que les propriétés élémentaires des matrices orthonormées, on obtient que :

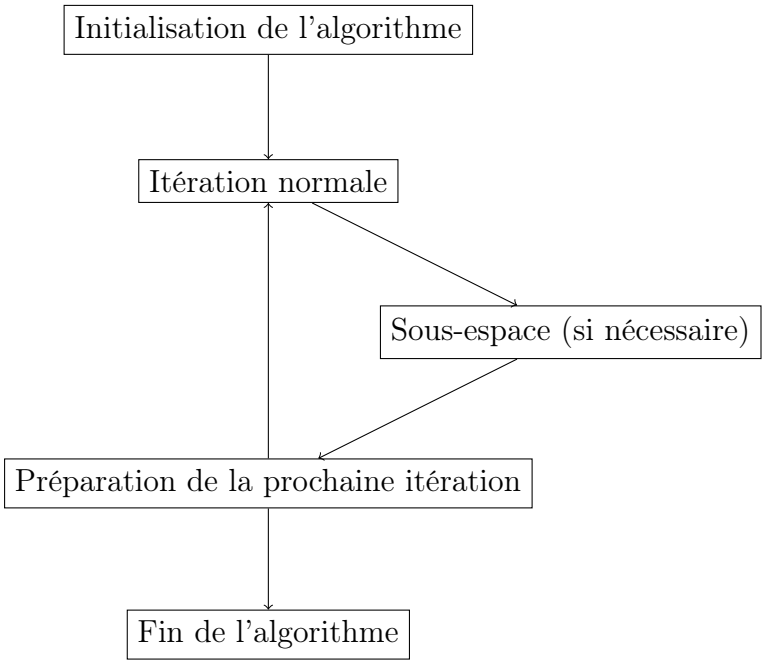
$$d_{k+1} = -Z(\hat{P}_k - \sigma_k \mathbb{I})\hat{g}_{k+1} - \sigma_k g_{k+1} + \beta_k^+ d_k,$$

$$\beta_k = \sigma_k \left(\frac{y_k^t g_{k+1} - \hat{y}^t \hat{g}_{k+1}}{d_k^t y_k} - \left[\frac{||y_k||^2 - ||\hat{y}_k||^2}{d_k^t y_k} \right] \frac{d_k^t g_{kk+1}}{d_k^t y_k} \right),$$

$$\eta_k = \eta\Big(\frac{s_k^t g_k}{d_k^t y_k}\Big),$$

$$\beta_k^+ = max(\eta_k, \beta_k).$$

Nous pouvons donc visualiser l'algorithme de la manière suivante :



7.1.5 Calcul sans reconstituer explicitement Z

Une difficulté de cet algorithme serait de trouver la base orthornomée Z de S_k. Cependant, les auteurs mentionnent qu'une mise à jour de rang un peut être faite sur une décomposition QR de S_k. Par efficacité de calcul, on remplace la matrice Z par $S_k R^{-1}$ à chaque fois qu'elle est nécessaire. Il ne reste donc qu'à mettre R à jour et à l'inverser. Or, m (le nombre de directions gardées en mémoires) est petit par rapport à n (la dimensionnalité de notre problème), et R est triangulaire. L'économie serait donc substantielle et rendrait la méthode fortement compétitive. Comme la mémoire n'est changée que d'un vecteur par itération, il est simple de faire une mise à jour de rang un de type

$S_{k+1} = S_k + uv^t$. Pour se faire, on pose $u = S_{k+1}[:, i] - S_k[:, i]$ et v un vecteur de zéros, sauf à la i^e position.

7.1.6 Résultats préliminaires

La dernière version implémentée est en effet plus économique en terme de ressources utilisées, mais nous n'atteignons pas encore l'augmentation des performances escomptée, et ce, surtout par rapport au temps de calcul. De plus, certains artefacts apparaissent dans les images reconstruites et affectent fortement le rendu des images finales. Un examen accru du code libre d'accès en C fournit par Hager et Zhang pourrait fournir à un lecteur motivé les réponses au fonctionnement décevant de cet algorithme.

ANNEXE B

Nous incluons ici les images n'ayant pas été sélectionnées pour l'ouvrage principal. Selon nous, certaines de ces images alourdissaient le texte et nous avons jugé bon de simplement les mettre en Annexe. Lorsque ces images sont issues d'un test du chapitre cinq, le même titre de section est donné et les images manquantes y sont ajoutées.

8.1 Utilisation d'un a priori anatomique pour l'algorithme MLEM

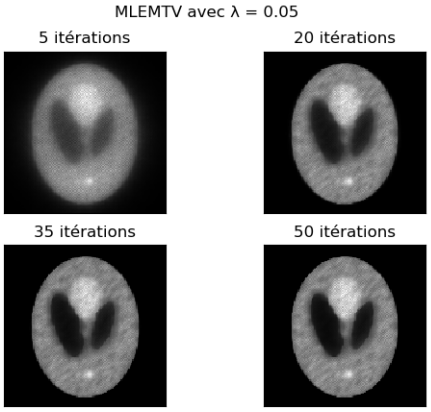


Figure 8.19 – Images reconstruites par MLEM avec $\lambda = 0.05$

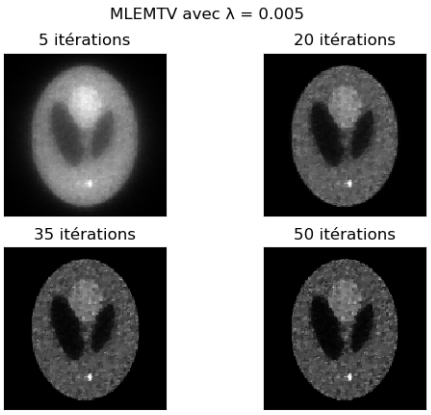


Figure 8.20 – Images reconstruites par MLEM avec $\lambda = 0.005$

8.2 Sélection du représentant de la famille des gradients conjugués

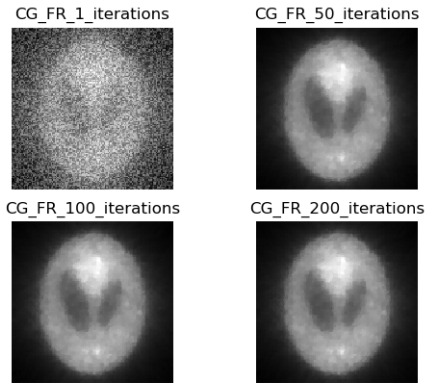


Figure 8.21 – Images reconstruites par CG_FR

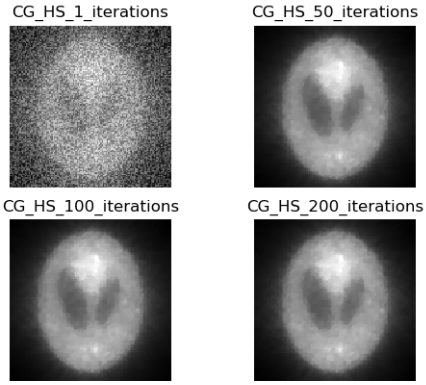


Figure 8.22 – Images reconstruites par CG_HS

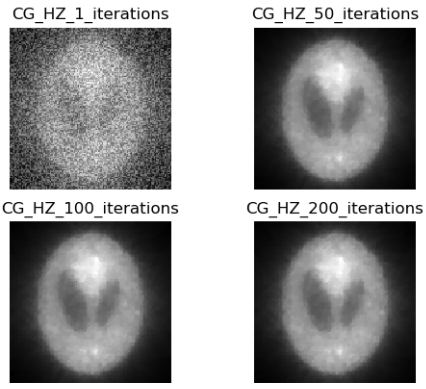


Figure 8.23 – Images reconstruites par CG_HZ

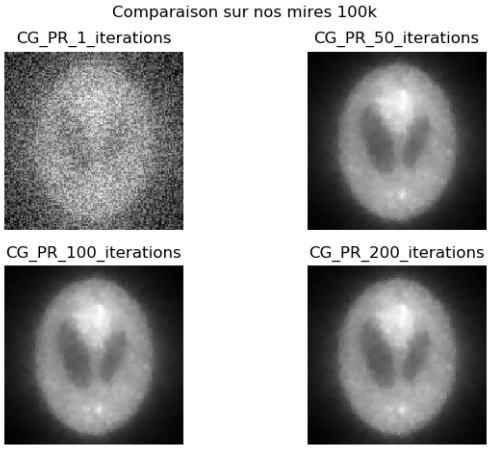


Figure 8.24 – Images reconstruites par CG_PR

8.3 Barrière logarithmique et algorithmes ne gérant pas les bornes

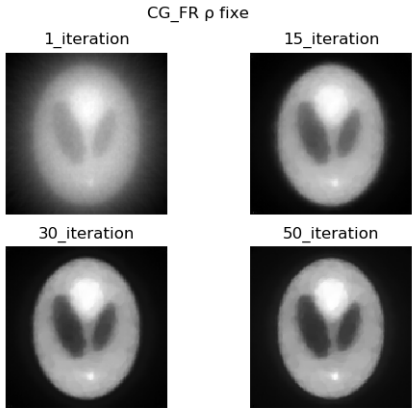


Figure 8.25 – Images reconstruites par CG_FR avec un ρ fixe

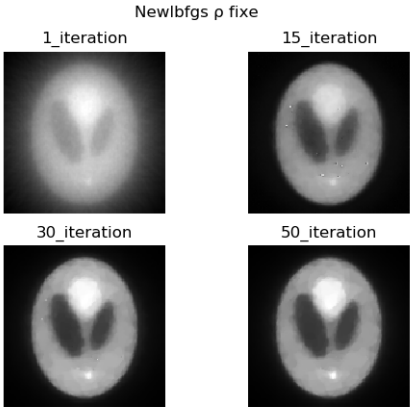


Figure 8.26 – Images reconstruites par L-BFGS avec un ρ fixe

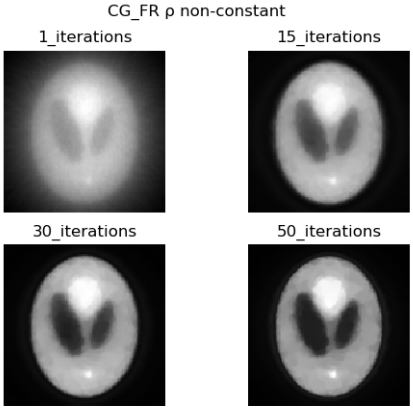


Figure 8.27 – Images reconstruites par CG_FR avec un ρ adaptatif

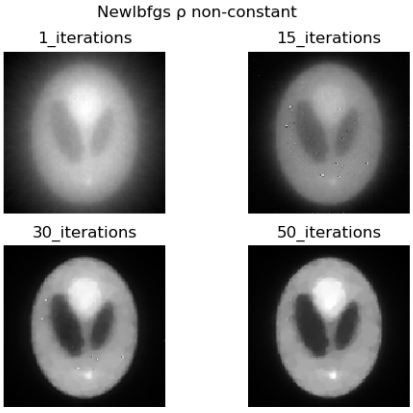


Figure 8.28 – Images reconstruites par L-BFGS avec un ρ adaptatif

8.4 Effet d'une meilleure initialisation

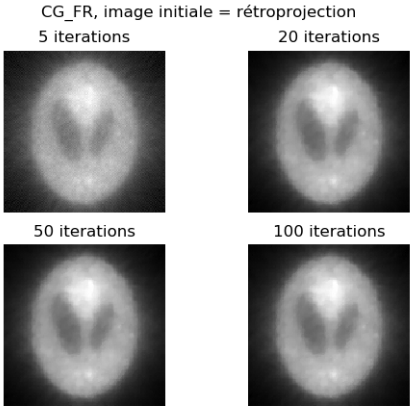


Figure 8.29 – CG_FR initialisé avec une rétroprojection

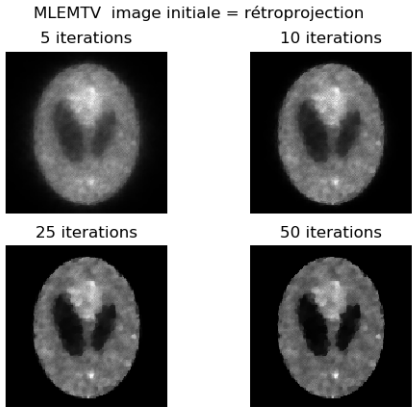


Figure 8.30 – MLEMTV initialisé avec une rétroprojection

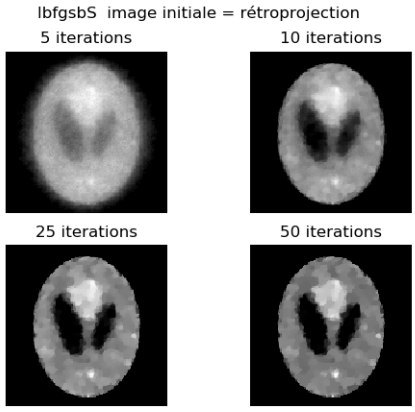


Figure 8.31 – L-BFGS-B initialisé avec une rétroprojection

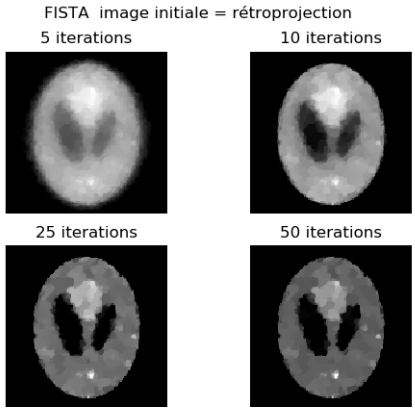


Figure 8.32 – FISTA-FGP initialisé avec une rétroprojection

8.5 Reconstruction LS en TEP

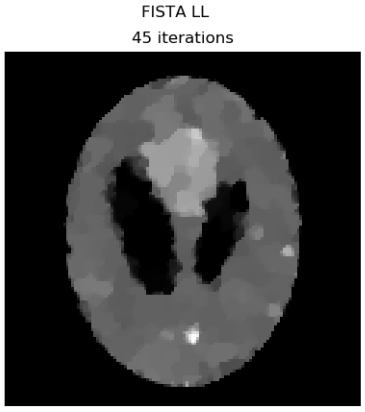


Figure 8.33 – Images reconstruites par FISTA-FGP avec la vraisemblance logarithmique

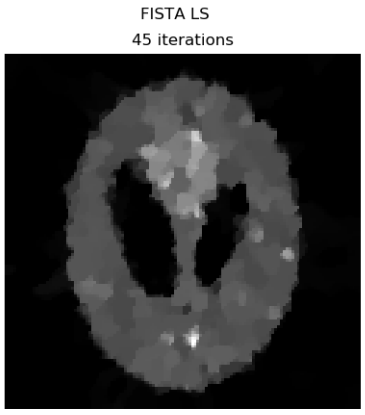


Figure 8.34 – Images reconstruites par FISTA-FGP avec les moindres carrés

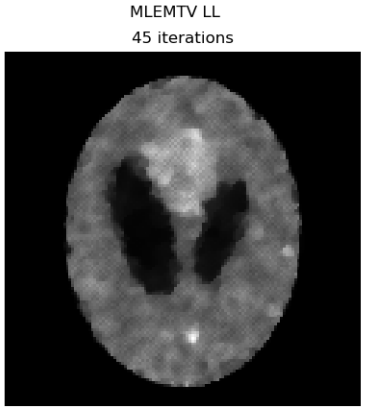


Figure 8.35 – Images reconstruites par MLEMTV avec la vraisemblance logarithmique

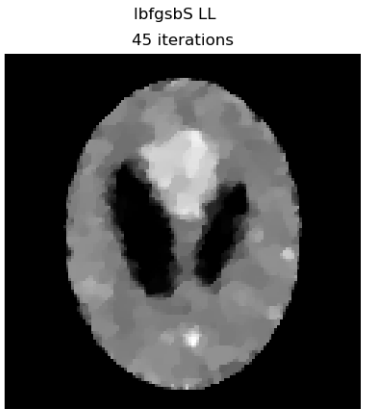


Figure 8.36 – Images reconstruites par L-BFGS-B avec la vraisemblance logarithmique

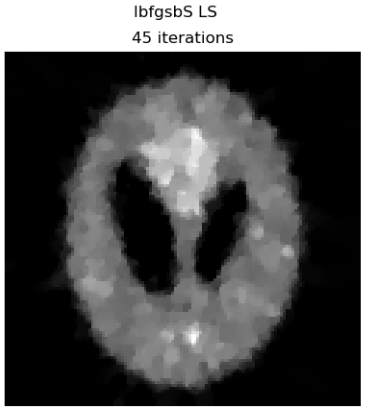


Figure 8.37 – Images reconstruites par L-BFGS-B avec les moindres carrés

8.6 Reconstruction LS en TDM

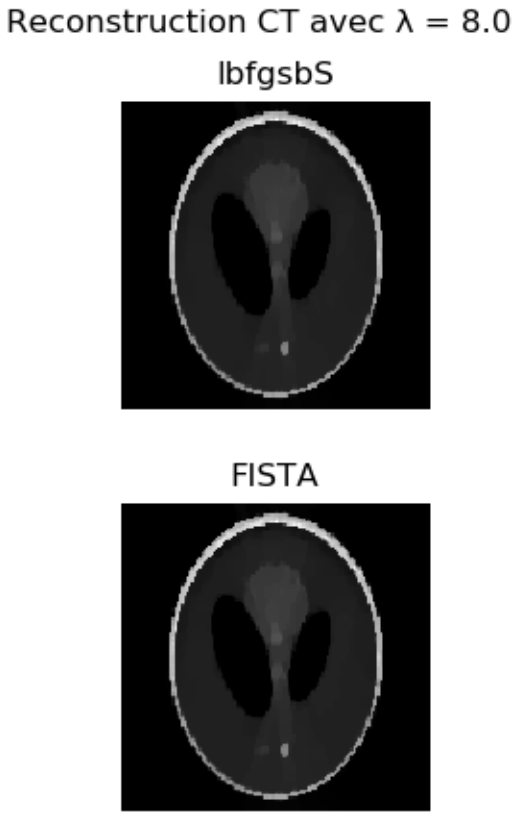


Figure 8.38 – Image reconstruite par L-BFGS-B et FISTA-FGP avec les moindres carrés en tomodensitométrie

Bibliographie

[AEAF01] Idris A Elbakri and Jeffrey A. Fessler. Statistical x-ray-computed tomography image reconstruction with beam- hardening correction. 04 2001.

[Ass90] I. Assem. Tilting Theory – An Introduction. In *Topics in Algebra*, volume 26 Part 1 of *Banach Center Publications*, pages 127–180. PWN Polish Scientific Publishers, Warsaw, 1990.

[BB88] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1) :141–148, 1988.

[BKSE12] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia : A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[BLNZ95] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5) :1190–1208, 1995.

[Bro70] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1) :76–90, 1970.

[BT09a] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11) :2419–2434, Nov 2009.

[BT09b] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1) :183–202, 2009.

[Cau47] A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *C. R. Acad. Sci. Paries, 25*, pages 536–538, 1847.

[DHL17] Iain Dunning, Joey Huchette, and Miles Lubin. Jump : A modeling language for mathematical optimization. *SIAM Review*, 59(2) :295–320, 2017.

[Dus17] Jean-Pierre Dussault. *Optimisation mathématique avec applications en imagerie*. Université de Sherbrooke, 2017.

[Fle70] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3) :317–322, 1970.

[FR64] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2) :149–154, 1964.

[Gol70] Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109) :23–26, 1970.

[Ham09] Benoit Hamelin. *Accélération d'une approche régularisée de reconstruction en tomographie à rayons X avec réduction des artéfacts métalliques*. PhD thesis, École Polytechnique de Montréal, 2009.

[HGD] Benoit Hamelin, Yves Goussard, and Jean-Pierre Dussault. Comparison of optimization techniques for regularized statistical reconstruction in x-ray tomography. *2010 2nd International Conference on Image Processing Theory, Tools and Applications, IPTA 2010*.

[HJ11] Per Christian Hansen and Jakob Heide Jørgensen. Total variation and tomographic imaging from projections. In *Proceedings of WSC 2011*, 2011.

[HL89] T.J. Hebert and Richard Leahy. A general EM algorithm for 3-D Bayesian Reconstruction from Poisson data using Gibbs priors. 1989.

[HS52] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems, Volume 49*. Number 1. NBS Washington, DC, 1952.

[HZ13] W. Hager and H. Zhang. The limited memory conjugate gradient method. *SIAM Journal on Optimization*, 23(4) :2150–2168, 2013.

[JB18] Jakob Sauer Jorgensen William Lionheart Joost Batenburg, Per Christian Hansen. *Scientific Computing for Computed Tomography*. Université technique du Danemark (DTU), 2018.

[Kad01] Dan Kadrmas. Statistically regulated and adaptive EM reconstruction for emission computed tomography. Nuclear Science, IEEE Transactions on :790 – 798, 2001.

[Kau93] Linda Kaufman. Maximum likelihood, least squares, and penalized least squares for pet. *IEEE transactions on medical imaging*, 1993.

[LTK+11] L. Lu, J. Tang, N. Karakatsanis, W. Chen, and A. Rahmim. Cluster-based priors for map pet image reconstruction. In *2011 IEEE Nuclear Science Symposium Conference Record*, pages 2678–2681, Oct 2011.

[Nes83a] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate o(1/k$\hat{2}$). *Dokl. Akad. Nauk SSSR*, 269 :543–547, 1983.

[Nes83b] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 1983.

[Nes14] Yurii Nesterov. *Introductory Lectures on Convex Optimization : A Basic Course*. Springer Publishing Company, Incorporated, First edition, 2014.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, USA, second edition, 2006.

[NY83] A.S. Nemirovski and D.B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983.

[Pow77] M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, pages 241–254, 1977.

[Rad86] J. Radon. On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging*, 5(4) :170–176, 1986.

[ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 1992.

[Sha70] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 1970.

[Sim15] C. Simard. *Analyse d'algorithmes de type Nesterov et leurs applications à l'imagerie numérique*. Université de Sherbrooke, 2015.

[SV82] L. A. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, 1(2) :113–122, Oct 1982.

[TBE+18] Y. J. Tsai, A. Bousse, M. J. Ehrhardt, C. W. Stearns, S. Ahn, B. F. Hutton, S. Arridge, and K. Thielemans. Fast Quasi-Newton algorithms for penalized reconstruction in emission tomography and further improvements via preconditioning. *IEEE Transactions on Medical Imaging*, pages 1000–1010, 2018.

[TBZ16] S. Tao, D. Boley, and S. Zhang. Local linear convergence of ista and fista on the lasso problem. *SIAM Journal on Optimization*, 26(1) :313–336, 2016.